Faisal Usman

Development of a User Centered Based user interface for building automation control

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 28.12.2018

Thesis supervisor:

Prof. Jaakko Ketomäki

Thesis advisor:

Mika Maaspuro, Lic.Sc (Tech.)

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Faisal Usman

Title: Development of a User Centered Based user interface for building automation control

Date: 28.12.2018 Language: English Number of pages:7+66

Department of Electrical Engineering and Automation

Professorship: Smart Living Environment Code: ELEC 3023

Supervisor: Prof. Jaakko Ketomäki

Advisor: Mika Maaspuro, Lic.Sc (Tech.)

In recent years, the field of smart buildings has generated much interest for indoor environment where people spent most of their time. Intense research has been focused on improving the building automation system (BAS) to incorporate smart tools and services in order to improve system efficiency and provide well responsive, safe and proactive indoor environment. However, little effort has been made in improving user building interaction that not only restricted the user's ability to easily operate the system in fact it has also lowered down the user's confidence for the system.

The first objective of the thesis was to select a better option of user building interaction type by doing review of previous research work and comparative analysis of existing systems in the market based on evaluation parameters. From the comparative analysis, it was concluded that only voice user interface (VUI) exhibits complete system acceptability in addition to the flexibility, portability, remote accessibility, usability in comparison to other user building interaction types. Moreover, Amazon Alexa is used for development of VUI that is more system adaptable than other VUIs. The second objective of the thesis was to develop and implement a UI prototype based on selected user building interaction mode to determine its feasibility of integration with existing BAS. A system architecture is proposed and four skills were designed that not only determine the feasibility of control of devices using voice but also capable of performing non-productive tasks on behalf of users to save time for them. The Amazon Alexa and Amazon Web Services (AWS) are primarily used as the development platform for interaction model and backend services for a User Interface (UI). Whereas third party APIs are also used to execute tasks outside of AWS cloud. The developed skills were tested and deployed in the system where multiple users can access the skills from multiple room profiles.

Keywords: Smart building, Indoor environment, Building automation system, User building interaction, User Interface, Voice user interface, Amazon Alexa, Amazon Web Services

Preface

First, I would like to thank Allah Almighty, for giving me the strength to fulfill this task. It would not have been possible without His will.

This thesis work has been carried out in 'Smart building technologies and services' research group, Aalto University. I would like to express my deepest gratitude to Prof. Jaakko Ketomäki and Prof. Heikki Ihasalo for giving me this wonderful opportunity to work in their research group. I would like to thank them for their ideas, advices and valuable suggestions regarding my thesis. I also want to thank my instructor Mika Maaspuro for his suggestions and corrections in the report writing and helping throughout my thesis.

I would also like to thank my friends Vladimir Kuliaev, Usama Riaz, Abid Saleem, Bilal Mustafa, Ammar Arshad and Yaseen Ahmed Khan for their constant moral support during my stay in Finland.

I would also like to express my deepest love and gratitude to my mother, Mrs. Farah Usman for her endless support, my gorgeous wife Dr. Amna Saeed for loving and encouraging me, my sisters Sana Abid and Dr. Hina Masham for being there for me at every step of life. I would also like to thank my close relatives especially Prof. Ghulam Haider Chishti, Saima Rizwan, Dr. Saeed Ahmad Sheikh, Dr. Uzma Saeed, Abid Shahzad and Masham Khalid for their kind prayers and continuous support at every difficult part of my life.

Lastly, I would like to dedicate my work to my daughter Mahzaib Faisal and to my late father Muhammad Usman whom dream was that I become an engineer.

Espoo, 28.12.2018

Faisal Usman

Contents

List of Figures

Abbreviations

UI	User Interface
EIBG	European Intelligent Building Group
BAS	Building Automation System
M2M	Machine to Machine
HDML	Handheld Markup Language
RFB	Remote Frame Buffer
API	Application Programming Interface
SMS	Short Message Service
VUI	Voice User Interface
URL	Uniform Resource Locator
AVS	Alexa Voice Service
STT	Speech-to-text
TTS	Text-to-speech
CLI	Command Line Interface
GUI	Graphical User Interface
WWW	World Wide Web
HVAC	Heating, Ventilation and Air Conditioning
AWS	Amazon Web Services
IOT	Internet of Things
A4B	Alexa for Business
TSL	Transport Security Layer
HTTPS	Hyper Text Transfer Protocol Secure
SSL	Secure Sockets Layer
TLS	Transport Layer Security
PaaS	Platform as a Service
MQTT	Message Queuing Telemetry Transport
JSON	JavaScript Object Notation

1 Introduction

1.1 Background

Smart Buildings have become increasingly incorporated in daily life and especially it has become a vital factor of productivity at workplace. People mostly spend their time in indoor, they want their indoor environment to be well responsive, safe and secure, energy efficient, and proactive. It will increase their productivity, ensure health, wellbeing and safety. Advances in Smart tools and services have attracted considerable attention of organizations in recent decades. In last few years, the smart building has been the focus of most researchers that leads to product development.

Generally, building automation and user building interaction are two major components of the smart building. Development of control, communication and dynamics of building appliances had been the focus of intense research whereas little work had focused on user interaction with building appliances. With the rapid advancement in building automation, it became extremely important that user interaction with building appliances should not only be accessible but also be useful to user thereby improves system's performance and user's usability.

In last decade, significant improvement have been made in the field of building automation. Initially, the defined intelligent building was technology centered based and did not put much attention to user interaction. There were some criticisms that it failed to survive with the change in building's occupancy, environment and information technology. Some authors [1, 2] suggested that intelligent building should respond to users' requirements. Then modified modern definition also focused on the user centered approach to the intelligent buildings in addition to automation. According to European Intelligent Building Group (EIBG), a created building should provide comfortable environment for its users and efficiently managing the resources at the same time [3].

With the modern definition, it enabled the cooperation of other building systems with building automation; the smart building concept was introduced which referred to the built environment. It allowed the cooperation of smart devices with user in-

teraction that include both frequent as well as sporadic visitors with the building control system. The concept of the integrated technological system was introduced to create a building that provides user with effective, flexible, secure and comfortable environment. However, the interaction between a user and a building achieves through different user interfaces (UI).

The interface between the building control system and user depicts the method and level of control that user has over his/her building operations. A well-developed UI has a positive impact on user experience with the building control and improved user's satisfaction [4]. Improved user satisfaction means increased user productivity [5, 6]. There are different UIs available like mobile apps, dedicated touch screens, keypad/ handheld remotes, web applications, voice UI and hand gesture based UI that can be used for building control. Every UI has its own merits and demerits depending upon the cost, operation, usability, user compatibility and user comfort. It is very important that a user feels comfortable while operating the UI.

User experience is a specific domain that concerns the usefulness of UI and user's satisfaction towards UI product. There are two sections of user experience: user interface design and user interaction design. Both these are related to each other. User interface design is directly concerned with the user, as interface design will be interacted with the user. However, user interaction design defines the behavior of interaction between user and the system. A good interaction design should be able to effectively communicate the system´s interactivity and functionality, define behaviors to user interaction, define simple and complex work–flow and inform the user about changes and avoid and prevent user errors.

The most natural source of interaction for human is the voice. However, for user building interaction, this certainly is a difficult task for understanding the user [7] . Nevertheless, the recent invention of virtual assistant technology has opened up new horizons for user system interaction especially with building systems by integrating voice UI with Building Automation System (BAS). Some notable devices that recently introduced include Google Smart Home and Amazon Alexa. The voice recognition has certainly eased for elder user to save time that spent on familiarizing with operating technologies and for users with disabilities to use the smart building services [8]. In addition, virtual assistants are very popular in assisting the user in less productive work especially in workplace. This saves time for employees at workplace so that they can concentrate more on high valued tasks, thereby certainly can help in improving their productivity and morale.

1.2 Motivation

Although these researches have made significant improvements in building automation components specifically in control and the machine to machine (M2M) communication of building appliances, few studies has focused on the improvement of user building interaction. Unfortunately, this gap between advancement of other

building components and user building interaction has not only limited the user's ability to operate the system but also lowered down the user's confidence for the system. Therefore, one potential approach for addressing this issue would be integrate UI design approach with smart buildings that must be capable of assisting user efficiently and help them in increasing their productivity especially at workplace.

1.3 Research Aims

Thus, the aim of this thesis is to develop a working prototype of the user centered based user interface that determines its feasibility of integration with a building automation system.

To accomplish this task, this thesis will make comparative analysis based on literature related to usability analysis and technical abilities on different types of UI designs available in market and their mode of user building interaction. Based on comparative analysis, the best possible user building interaction mode will be selected and UI will be developed in a way to provide user more usability and usefulness.

The scope of this thesis work will be limited to the development of UI for user building interaction. The idea is to study different UIs and their user interaction modes, their market impact and pros and cons and select the best possible user interaction mode that can integrate with BAS of the Pilot Project, Aalto University. The 'Pilot Project for smart building services technology (Pilot Project)' is an Aalto University project with the aim to investigate the behavior of building towards improved efficient workspace by implementing smart solutions. The one such smart solution is to implement new modes of user building interaction that can improve both user's satisfaction and efficiently maintain building operations [9].

In view of the scope, the main outcomes of this thesis work consist of:

- Selection of the best possible user building interaction type based on comparative analysis
- Development of UI prototype that can integrate with the existing BAS using cloud to cloud communication

2 Literature Review

This chapter presents detailed literature review related to different user building interactions and UIs based on their evolution, merits/demerits and their use in existing systems. It also introduces the evaluation parameters for UIs and evaluates UI designs of existing systems. In addition, it also explores the operating principles of selected UIs. Sections 2.1,2.2,2.3 and 2.4 describe work related to different user building interactions, introduction of evaluation parameters, types of UI designs and comparative analysis of existing systems whereas Section 2.6 and 2.7 explores voice user interface and virtual assistants respectively.

2.1 Related Work

UIs are very important for any BAS performance. It allows the user to take control of the system and get familiar with the system functionality. With rapid advancement in BAS functionalities, there was a need to develop an efficient UI that can interact with the advanced building control system and at the same time, allow user to get familiar with the system. In order to evaluate different UIs, it is necessary to present previous research work related UIs used in different proposed BAS. Thereby, this section reviews the work done in the field BAS related to UI.

Initially, either dedicated buttons or dedicated screens are developed for each machine and connected with the machine. These UIs mainly operated by expert users that excluded end users. The other drawback was their inability to be used for any other system therefore they provided limited functionality and lacked usability for end users, remote access, adaptability and accessibility as well.

The remote control device was developed and studied for control of various electronic devices. It was suggested that it can operate different electronic devices by adding more functions and memory components. Several methods have been applied to enable the remote control based UI for BAS. One such approach was proposed in [10], where the microcontroller based remote device with plug in memory used to control various devices' functions connected in BAS. The author also included an

additional display element for informing the response of the system based on the user input. It allowed the UI to remotely access the control system but communication of this device was based on Infra-red which has distance limitation from the IR receiver.

In [11], author proposed the menu driven based UI that particularly allowed menus to be distributed throughout a home. Due to this work, Menu driven interfaces provided different systems connected to the single and dedicated processor. These screens provided the same control function for all distributed locations within the environment. Therefore, the user would experience the same control from every available screen connected with the central processor. Generally, the menu driven based UI allowed series of screens to get navigated from available list options. Due to its simplicity, it mostly used for the walk-up-and-use systems. The user does not require any prior training to use system effectively. The menu driven based UI turned out be simple and easy to use solution and it does not require any commands. This certainly helped usability of UIs for BAS. Other than this, it also made the software portable and it could be usable with either personal computer, touch screen, remote control, home distributed networks, infra-red and television receivers.

Later in 1996, Michael Stein and Toby Ray Kaufman [12] suggested some improvements were made related to Menu driven based UIs for BAS. They introduced the conventional mechanical keypad with touch screen to enhance the proximity between system response and user commands as well as making it the cost effective.

In early part of the 2000s, different software language frameworks were discussed to enhance portability of UIs [13]. These technologies included Handheld Device Markup Language (HDML), Embedded Java and Remote Frame Buffer (RFP). Depending upon the technology used, the user can remotely control home appliances within the home environment by getting access from Web browser on Home PC, TV set and remote control, Java phone terminal and a personal organizer connected to home network using a wireless link. Though their prototype was developed using Embedded Java. After the introduction of Internet, the dependency of UI on hardware minimized and the users were able to access the building automation system using cloud services. Software can be designed either by using Web services or any Application Programming Interface (API). Authors [14] developed the web based UI by using Java whereas in [15], authors JAVA API used for development of the UI. The former proposed solution that enabled UI to work on mobile phone and user can remotely access the home automation system using the cellular network.

The above mentioned approaches offer remote access and software portability to some extent however failed to provide adequate information about system security and user privacy. By time, internet became extremely vulnerable due to different types of traffic, a lot of effort was required to keep the web based application secured. It would require resources like the dedicated hosting server, the uninterrupted internet connection. In view of this argument, it was emphasized to have mobile telephony instead of internet for interaction between user and home control system and communication was based on Short Message Service (SMS) [16]. This

approach allowed the system to exhibit some privacy but somehow excluded the software portability and made the system difficult for user to learn, memorize and effectively use it, therefore the system became less useful for the user. Though some efforts have been made to keep the internet safe and research is still in progress to make more advancements. Internet Security is the field dedicated for this purpose. So far, there are a number of firewalls and architecture that have been developed and implemented. For BAS, these techniques can be utilized for Web based services. In [17], the author adopted one approach by using IP based Architecture with home gateway system that consist of Internet Firewall and user interface was developed using RESTful based Web Services The designed framework can be accessed through any of Android device.

Since the advent of Voice as input, there are a number of applications that were discussed where the voice can be used as user input with good accuracy and the system do the hard work after understanding the input command from user. In 2003, Nielsen Norman Group [18] assessed the potential of the voice as UI and suggested that voice would help improve the accessibility of the system especially accessible for elder people.

In this context, some authors suggested that accuracy in speech recognition can be improved if noise level remains at certain level and small vocabulary should be used with limited grammar. They implemented it by integrating the voice based UI with the home device using the speech recognizer that converts speech into text commands and sends them to X.10 protocol for output. With the limited grammar, it certainly restricted the command format however it can be easily implementable if number of devices would be limited [19]. In [20], the proposed system used Microsoft API for speech recognition as well as text to speech feature. The motivation of implementing this system is to provide comfortable environment for elder people so that they can use minimum physical effort while interacting building systems. In this system, Microsoft API used to compare incoming speech with its predefined dictionary. This proposal certainly showed better results than previous systems.

In [21], G. Muthuselvi and Saravanan B used HM2007 voice recognition system for the UI. It had ability to store 20 samples of each 1.92 seconds speech. HM2007 compares the input voice with available commands database and then sends the address of particular command to the controller. It helps the system to become more real time and more responsive. Another voice controlled based automation system has been designed using Raspberry PI as a controller with the microphone and camera as input devices [22]. This controller designed to trigger on a keyword from user command and sends signal to the device controller.

Virtual Assistant is also used as a voice user interface (VUI) for automation systems. Virtual Assistant belongs to Artificial Intelligence and is capable of assisting user in daily routine work. In [8], authors used Alexa voice services for voice recognition with their developed voice user interface to sends commands to Raspberry PI for actions through local Ngrok secured Uniform Resource Locator (URL). Ngrok is a multiplatform tunneling that establishes the secure network from public endpoint

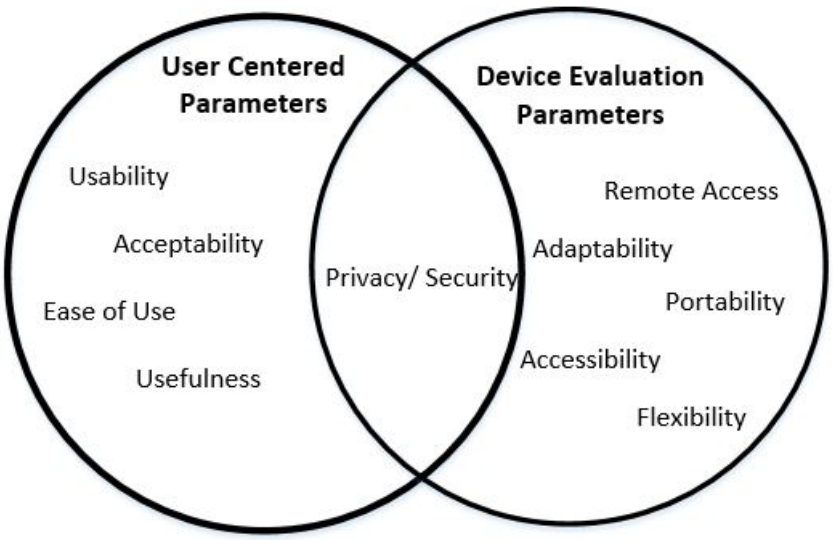


Figure 1: Two sets of evaluation parameters
a) User Centered Based Parameters b) Device Evaluation Parameters

to the local network. Another virtual assistant based voice user interface has been developed using Alexa voice services (AVS) and Raspberry PI3 with speech-to-text (STT) and text to speech (TTS) engines [23].

VUI has enabled elder and disabled people to effectively use the system and feel comfortable about the system. According to the research, it is adamant that technology supports this idea but most of the research mentioned only justified the opportunity of using the VUI for the smart building. In order to increase the usability of user, the UI should surely be flexible, portable and secure.

2.2 Evaluation Parameters

There are certain aspects that determine the quality of interaction between the UI and user specifically in the building automation. These aspects certainly connect technology with building and user and vice versa. These aspects are either evaluate the relation of technology with the system or technology with the user. It helps building a system which is convenient for user to operate. In this thesis, these aspects termed as evaluation parameters. This section briefly introduces the evaluation parameters that were used to evaluate UIs of existing systems. These evaluation parameters are:

1. Usability
2. Ease of Use
3. Usefulness

4. Acceptability
5. Remote Access
6. Portability
7. Accessibility
8. Privacy
9. Flexibility
10. Adaptability

These evaluation parameters are further divided into two sets as shown in Figure 1. The UI is itself an individual system which has its own controller that sends input parameters to the main system based on input received from user. Therefore, above mentioned parameters become extremely viable for evaluation of the UI as a system.

Usability:

Usability usually addresses the interaction quality between user and interface. It inspect several factors for evaluation. These factors mainly determine how much time being taken for completion of the task, how many errors were made and how much time user took to learn the system. It assesses the UI that how easy was it for user to use it. The attributes of usability are:

- Learnability
- Efficiency
- Memorability
- Errors
- Satisfaction

Usability is important the parameter for UI evaluation as it will tell the user's satisfaction with the system.

Ease of Use:

It is a measurement of easiness of an intended user while using the product. It is a degree that how much the user believes that it would be free of effort to use the particular system. It belongs to user's satisfaction that translates into product rating. It belongs to User Centered Set and can be obtained using empirical analysis.

Usefulness:

Basically, it is a degree to which the user believes that using a system enhances his job performance. Usefulness is slightly different from ease of use though it also a subjective property. It also belongs to User Centered Set and can be obtained using empirical analysis. It is an important aspect that can be related to user's performance results.

Acceptability:

It is an important aspect that tells whether user accepted the quality of UI or not in terms of either social acceptability or practical acceptability. It is a degree of satisfaction related to the user's needs and requirements from a system. In terms of practical acceptability, it mainly tells that either customer satisfied with cost, reliability or compatibility of the system or not. It also belongs to User Centered Set and can be obtained using empirical analysis.

Remote Access:

Remote access belongs to device functionality. This function allowed the device to be accessed from other locations. It is important aspect for UI as it gives user an option to control the system from offsite location or even away from control board of a machine. With the introduction of internet, networking and other communication protocol, it becomes a must available functionality of UI.

Portability:

Portability is the ability of a system to use it data through multiple software and hardware environment. Portability mainly concerned with software application which tells that either particular application can be run in different devices. It belongs to functionality of a system.

Accessibility:

Accessibility depends upon the target user group which means that what kind of user can be able to operate the system. Basically, it tells how a system is accessible to different users in terms of age, psychological ability, and physical ability. Accessibility mainly concerned with visual and audible modality. Audible modality provides extra incentive to old age, disable people especially those with visually irreparable and patients as well. It works as personal assistant to a user. Therefore, it creates a sense of ownership in user mind.

Privacy/ Security:

Privacy is a very important aspect of a system. It can be either categorized as user centered based but it is foremost important for a system to provide the security aspect especially if a system is remotely accessible. Therefore, it is placed within functionality group.

Flexibility:

Flexibility is a feature of the device that give the user freedom to customize some its setting like in case of the user interface, change the display color, theme etc. But particularly for a BAS, it either allows user to customize operational settings or not.

Adaptability

Adaptability describes whether a one UI can be used with multiple systems or the only dedicated one system. Other than this, it also evaluates whether the system

is adaptable to the changes made by use via UI. Adaptability is the important functionality of a system because it maximizes the system's operability.

2.3 Types of UI Designs

There have been different types of UI design invented and used in the market. It ranges from Command Line interface to Gesture Driven.

Command Line Interface (CLI):

It requires specific commands for operation. It has its own syntax. It is basically user's responsibility to learn the commands and how to phrase those commands with their parameters. CLI is still a very powerful tool as it is a base of some very popular operating systems that includes MS-DOS, Linux and UNIX. It gives significant user control over system's operation. In terms of user especially end user, one must get oneself trained as command interfaces are tough to learn. Generally, people just learn a few commands in any command based language and make mistakes in command entry rather more frequently. It requires a big cost for training. Though corrections can be made but still it is also costly for users and companies and result in loss of productive time.

For BAS end users, it may take a lot of time to get familiar with CLI based UI.

Menu based interface:

This type is simple and easy to use. It does not require any commands. The options are given in step by step order therefore user doesn't have to remember anything. Menus can become too slow at time when it is being used to analyze complex situation, consumes a lot of time and take several menus to operate a system. It irritates the user while navigating for a required option from too many options therefore sometimes it becomes hard for user to read. However, menus can often be effective as well because they rely on recognition instead of recall. While working menus, user get familiar with the available and it gets easy to operate.

A graphical icon's based tool bar can also be categorized as a menu. But the designer must take into consider difference between inexperienced and experienced users.

Graphical User Interface (GUI):

It is an interface where user have direct control of visible objects. It is a most common used user interface. Windows operating system, Macintosh operating system and World Wide Web (WWW) are best examples of GUI. It contains pictures, icons and graphics and make a friendly display for users. Users point and click to make actions instead of entering commands. It is also called WIMP Interface as it mainly consists of Windows, Icons, Menus and Pointers.

GUI is very easy to use and explore. It doesn't require complicated commands. It usually includes "Help System" with WIMP Interface that provides more convenience

and support to users. Technically, it occupies much larger hard disk space than Menu based UI and CLI as it requires extra processing power to operate. Regardless of technical limitation, it provides user more easiness and simplicity than Menu based UI and CLI.

Natural Language Interfaces

This type of interface allows the user to make input either by speech or text. The input is generally based on user's everyday normal language. There are three types:

1. Voice User Interface
2. Form based User Interface
3. Gesture based User Interface

For a VUI, the speech recognition system is used to recognize voice input and convert it into text command that is send to the controller for action.

Other type of natural language interfaces is to use the text box like a chat box where a system will interpret your questions, then return a suitable and relevant answer to the question.

Third type of natural interface is a gesture based UI. Gesture interaction is relatively a new concept. It is associated to a response with user's intention of achieving a result. The gesture is a non-verbal communication.

The computer identifies a gesture by using imaging data that represent the gesture. This technique is relatively more technical and complex for the developer to design. Its challenges are mainly related to accuracy and usefulness. This system has to be accurate enough to understand the gesture without considerable noise.

Natural Language Interface is challenging to designers in comparison to other interfaces. This type of interface has to understand the intention of user and sends commands accordingly. It requires a system to work more for user convenience. They have limitation in terms of device and intelligence level. They should be enough intelligent to understand the user's intention.

2.4 Comparative analysis of existing systems

The BAS has been rising significantly in the last decade. The market of BAS has been advancing every year with each year an increase of demands to implement an integrated system. During these years, various companies have also improved their solutions and in the meantime, a large number of new ventures has been started with a vision to provide services in the smart building. Hence companies certainly improved their BASs therefore it will be interesting to analyze their work related UI designs.

This section will evaluate UIs of some of existing systems. The idea was to assess the operational capability of UI whether it is compatible to system operations based on functionality parameters. Five existing systems were chosen based on the search term "most popular building automation systems" using google search engine. These five building automation systems include Crestron, Loxone, Control 4, Zipato and Vivint.

Evaluation method uses three sets of evaluation parameters, these sets are system features, user adaptability and functionality. 'System features' consist of five features; Lighting, Heating, Ventilation and Air Conditioning (HVAC), Security, Music and Window shading. User adaptability consists of flexibility, privacy and UI customization. Whereas functionality will be assessed based on features of remote access, system adaptability, portability and accessibility.

1. **Loxone:**
   Loxone was designed thinking of all devices that a building or a room can possibly have. This system is capable of controlling Lighting, HVAC, security system, Music and window shading. It fulfills all the criteria for system features.
   In terms of a UI, it has also provided flexibility for users by allowing to do scheduling, make different scenes and pre-set scenarios that can automatically be initiated if certain conditions met in addition to normal operation. Alternatively, it also gives an option to the user to customize.
   The user interface app is available in iOS, Android and Web Browser. This feature gives user a freedom in terms of choosing a operating system according to his social acceptability towards the system. This system is based on cloud services and provide complete security and privacy. It requires user authentication in terms of username and password. It uses IP connectivity to communicate with system cloud. In terms of visual modality, the UI has used metaphors, intuitive context. This system is adaptable to exhibit changes by UI. However, it does not provide voice recognition thereby lack complete accessibility in case of audible modality. With the voice input, there is no need to carry his device around the building. Once he is inside, he can make task request to the system. Voice recognition makes it happen and turns out to be an extremely valuable point for this evaluation.

2. **Control4:**
   Control4 supports all required system features that we looked for. It provided the same features as Loxone. It also possesses flexibility thereby allowing users to either schedule tasks, make custom scenes or pre-set scenarios. In terms of security, the system provides it through user authentication, the system requires first time login and connects through IP connectivity. It does not allow the user to customize UI. In terms of remote access, its main UI is a dedicated screen however it also provides limited functionality in IOS but it does not offer any android or web basedUI thereby lacks in software portability. The provided UI can be operable for only one system at a time although it

is also available in iOS. Perhaps this system is not adaptable. In terms of accessibility, it exhibits a contextual UI but the system does not allow to make changes in visual appearance. Other than this, Control 4 does not provide any voice recognition feature either therefore accessibility does not help much to cover all group of people.

3. **Zipato:**
   Zipato is the first one in the list that does not offer all the required features. Although it provides control for lighting, HVAC and security but it is not applicable to music and window shading. In addition to system features, it also slightly lacks in terms of user adaptability. It offers flexibility in task scheduling and custom scenes but there is no option for pre-set scenarios. Privacy is ensured by session based login and use of IP connectivity protocol for communication.
   The purpose of session based login is to work with a login system which is for the system to log the user's activity and provide options for managing the electricity consumption and help reduce energy related expenses. However, there is a restriction for UI customization. In terms of adaptability and portability, it exhibits the same situation as Control4 except its app is available in Android instead of iOS. In case of accessibility, there is not much consideration made as there is little to offer in visual modality as well as no voice assistant.

4. **Crestron:**
   Crestron has included most of the features that belongs to the smart building. This can possibly control all the system features that are listed thereby include lighting, HVAC, security, and window shading.
   Just like Zipato, Crestron exhibits session based login and does not allow for UI customization. Secured connection establishes using IP connectivity. However, it offers some flexibility in terms of custom scenes. Due to absence of session login,a single UI cannot be used for multiple building automation systems at a time. Therefore, it is not so adaptable to different systems. Perhaps, it also has the limitation in terms of portability because its UI is only available in iOS and it is unclear whether data can be transferred to other applications.
   Whereas it is the only system that offer both visual and audible modalities. It offers contextual GUI which allows the user to intuitively interact with the display. Other than this, it also offers voice assistant to the user, which ease the user burden in controlling the system manually thereby give the user freedom to control BAS while doing some other work.

5. **Vivint:**
   Vivint's home automation system is primarily based on security vision. Its Go!Control panel monitors all activities as well as provides time-based triggers and device activation announcements. User is given more control via its online control panel. In the web-based application, you can create and modify time-based and scripted triggers for all of your devices.
   Vivint offers something similar to session-based software. However, the disad-

vantages of the product are the inability to offer Software Integration and not allowing UI Customization. The system also provides mobile applications on several platforms that you can use to control your devices. Vivint does not offer voice recognition commands.

2.5 Results and Discussion

From the above discussion, a comparison is made and listed in Table 8 given in Appendix A. According to the table, all systems lack flexibility regarding UI customization. Except for Crestron, other systems have not provided UI customization feature. However, the users should be able to customize as they like to utilize maximum functionalities in a most efficient way. Some system also lacks session based login which is a weaker side of system security and user privacy. Most of the systems are remotely accessible.

Finally, there is only one system that provides voice recognition whereas all other systems do not allow voice commands. Voice recognition is an important feature as mentioned in Section 2.2 that will allow all age group with different abilities to interact with the system thereby improve the system adaptability. It is also easy to learn for all ages [19]. Voice recognition can be used for performing command or operating a device without using a keyboard, mouse or pressing any button [23]. Therefore, this thesis will use voice as user building interaction under consideration of evaluation parameters described in Section 2.2. Section 2.6 describes the voice based UI design.

2.6 Voice User Interface

Voice user interface (VUI) allows the user to interact with a system through either speech commands or voice. Voice interaction excludes the use of eyes, hands that help user to do other work in addition to interaction with the system. VUI does not provide any visual thereby user does not have a clear indication of the operation. It is important that user must be told about its functionality and limitation thereby clear indicate the interaction possibilities.

The biggest challenge for VUI is to understand the user intent from user's voice/speech.

Generally, there are four types of voice interaction.

1. Command Based
2. Speech to Text (STT) / Text to Speech (TTS)
3. Conversational
4. Personal Identification

In command based input, voice commands were given as input, a system recognizes the commands and act accordingly.

In STT, speech is processed into text and the text sent to the system controller. In TTS, text was given, the system understands and responds back in speech.

In personal identification, an outbound call asks user to identify himself by passphrase.

In conversation, user interact with the system through voice conversation, just like with another person. Voice conversation has low impact on cognitive abilities and help users to accomplish its task.

With a conversational interface, user can obtain information, access to web services, issue commands and engage in general chat by interacting in a normal way.

In [24], author discussed four components for conversational interface. These four components are:

- Speech recognition.
- Spoken language understanding.
- Dialog management
- Response generation
- text-to-speech synthesis.

Once the conversational interface has interpreted the input from the user, it constructs queries to Web services and knowledge sources in order to perform tasks and retrieve information to be output by the response generation component [24].

Speech recognition is a critical component of a conversational interface because it connects the user voice with the system and enables the user to interact with system operations. In [25], authors proposed three possible architectures to connect speech recognition with devices. These three architectures are:

1. Cloud voice recognition service and third party recording devices
2. Cloud voice recognition service and gateway as the voice recorder
3. Fully voice enabled gateway

2.7 Virtual Assistants

Conversational interfaces have become more popular due to recent development of virtual assistants. Virtual Assistants belongs to Artificial Intelligence field and have used deep learning that made improvements in speech recognition, spoken language understanding and dialog management. As a result of these technological advancements, user acceptance has increased for a voice conversational interface. Virtual Assistants also known as Virtual Personal Assistants, Intelligent Personal

Assistants, Voice Assistants, digital personal assistants, mobile assistants or voice assistants.

Virtual Assistants include: Apple's Siri, Google Now, Microsoft Cortana, Amazon Alexa, Samsung S Voice, Facebook's M, and Nuance Dragon.

The aim of Virtual assistants is to assist users to execute different tasks on their smartphones. These tasks can be either finding local restaurants, information gathering, getting directions, setting the alarm, updating the calendar or engagement in general conversation [24]. In addition to these tasks, virtual assistant has emerged as more accessible UI for controlling room devices. Their ability to act as convenient controllers for popular smart home devices makes them central figures in a smart home.

Apple's Siri, Google Home, Microsoft Cortana and Amazon Alexa are three most popular and preferable virtual assistants. They also provide APIs for control of building appliances and allow multiple user to incorporate within the same system using account authorization. These mentioned virtual assistants are all software agents that works on either on dedicated speakers or smartphones. The general working method is the same in above mentioned virtual assistants, they wake up using a specific keyword. The user input is recorded and deliver to the cloud server who has the ability to interpret voice input as a command. This cloud server than execute the action based on the user command like read back required information, play music or other tasks related to smart devices. With the action, the server sends the response back to voice assistant that informs the user with result.

Apple's Siri released in 2010 as standalone app has been longest around in the market, later it merged into iOS in 2011. In 2013, Microsoft released Cortana whereas Alexa launched in 2014 by the Amazon along Echo home speaker. Google Assistant was introduced in 2016 and it is available in both android app and home speakers [26].

The Amazon Alexa and Google Assistant are available both in app and on hardware platforms as home speakers. The Apple has recently announced Siri-enabled HomeKit device with HomePod as home speaker. The Amazon has released several variations in Echo products.Google Home Assitant also provide both mini and full size models. However, Microsoft is currently focusing on upgrading Cartona on a windows platform.

Today, most of the smartphones have voice assistants. Amazon Alexa is available both in Android and iOS. Whereas Google Assistant is available in Androids and Siri is only limited to iOS. Microsoft and Amazon are working together to bring Cortana to Amazon devices and Alexa to PCs. Whereas Apple's Siri is not compatible with any non-iOS device.

As the Amazon is the first to launch a complete home product with a large media library, the Amazon has become dominant player in a market for out of the box approach. Google is focused in increasing the capacity of Google products that

Table 2: Adaptability of Google Home, Apple Home Kit, Amazon's Alexa

Products	**Google Home**	**Apple Home Kit**	**Amazon's Alexa**
Smart Lights and Plugs	23	20	12
Smart Home Systems	14	15	18
Smart Thermostats	8	16	14
Smart Security Systems	10	-	18
Smart Locks	1	9	3
Smart Cameras	10	4	22
Smart Smoke/CO2 Detectors	1	3	2
Other Sensors	-	5	-
Cars and Car Accessories	4	4	11

can integrate with Google Home in addition to home based speaker, this approach certainly will rise its share in the market. Nevertheless, Apple with release of Siri-enabled HomeKit device with HomePod and other iOS compatible home connected devices, is also a strong member in the market. Whereas Microsoft is little slow in gaining market's attention as they lacked interest in developing home connected devices, therefore its market relatiely negligible.

All these devices are internet connected devices, each interaction sent to their device server that processes the user's command and provide the vistual assistant with a response. All these devices are compatible with IFTTT web services, which enable third party developers to design their skills and publish it.

Regarding privacy, Google uses voice printing, which uniquely identifies each user by voice and prevents the device form reading out personal information. Apple requires Apple ID to get access whereas Amazon's Alexa also requires user's voice for system access as Google does. So far, there are a number of products that have been developed that can work with these devices and additionally the same UI can be used for multiple products at a same time. The detail list of these features are given in appendix, however the summary of adaptability of Google Home, Apple Home Kit and Amazon's Alexa is given in Table 2.

Table 3: Overall comparison of Google's Home , Apple Home Kit and Amazon's Alexa

Categories	**Google Home**	**Apple Home Kit**	**Amazon's Alexa**
No. of devices supported	87	82	129
Working Mode	Online	Online	Online
IFTTT	✓	✓	✓
Multiple User Authentication	✓	✗	✓
Portability	✓	✗	✓
User Adaptability	✓	✓	✓
Privacy	Require user voice for system access	Require Apple ID	Require user voice for system access
Working Methodology	Dialogflow agent	HM Action Sets and Triggers	Alexa Skills

In view of above discussion, the overall comparison of Google's Home , Apple Home Kit and Amazon's Alexa is given in Table 3. In reference to overall comparison and device adapatability, Amazon's Alexa uses as VUI.

3 Amazon Alexa

Amazon Alexa is a virtual assistant that offers support to different services of daily life such as calendar logs, weather reports, emailing, playing music and home automation.

Alexa is a cloud based service that handles all the speech recognition, machine learning and natural language and understanding of all Alexa enabled devices [27]. Alexa has built in capabilities known as Skills, which define how a user can interact with the device. Third party integrators are supported using custom skills. Custom skill is a feature of Alexa develop by developers to integrate these systems and other internet connected services to AVS. Especially, home automation can be done by integrating it with smart systems like Google Nest, Phillips Hue, Smart Things and some other systems available in the market. Developers can avail the feature of Alexa Skill kit for customizing their program skills.

Other than AVS, Amazon echo with built-in AVS is also available which is personal smart speaker device and can be connected with other devices where internet connectivity is available. Users can either interact with an Amazon Echo device or a Mobile App by asking questions or making request and in turn, Alexa answers to the question or take action as per user desire.

Alexa skill is generally comprised of two elements and these are:

- Interaction Model
- Cloud based service

This chapter presents broad overview of technologies used for the prototyping of UI. Sections 3.1, 3.2 and 3.3 discusses the main components of Amazon Alexa that will be utilized in the development phase. The chapter also explores Alexa for Businees platform and AWS Internet of Things (AWS IOT) in detail in Sections 3.4 and 3.5.

Table 4: Difference between Interaction model and GUI

Action	**Interaction Model**	**GUI**
User request	User says, "Alexa, what is current temperature?"	User click a button
Collect more information	Alexa replies, "For what location" and waits for response	Dialog box open and wait for user to select an option
User provide more information	User replies, "Helsinki"	User selects options and click on OK
Complete user request	Alexa speaks, "The current temperature 1 degree celcius outside in helsinki"	App displays the result

3.1 Interaction Model

The interaction model actually defines what functionalities or behaviors the Alexa skill is able to handle. Interaction model will make a voice user interface and it is analogy to GUI. With this interaction model, user can interact with the app by using his/her voice instead of click and selection from dialog boxes [28]. The basic difference between the interaction model and GUI is presented in Table 3.1.

From this interaction model, Alexa interprets what user make requests or speak questions, translate into specific request that skill can handle. It is then sent to particular skill who has the capability to address the request or question. There are some built in interaction models available in Amazon Alexa developer dashboard like Smart Home skill API, Music Skill API and others. However, there is an option of customized custom skill for one's own interaction model.

Custom Skills:

Custom skills are a part of interaction model and can be designed using Alexa dashboard. When user interact with Alexa, for example, a request made by user is:

"Alexa, ask History master what happened on 14 August"

What custom skill does, it processes the request by parsing user's voice and send the request to cloud based service which is responsible to generate an action response. Action is then sent back to Alexa in a textual response. At the end, Alexa reads the textual response to the user. The whole process of custom skill needs several components to complete the requests. These components need to be created while designing a custom skill. These components are

1. **Invocation name:** Invocation name recognizes the skill. It should be included in the user's utterance when user starting conversation with the skill.

In the above example "History Master" is an invocation name, which instruct Alexa to invoke the 'History Master' skill. For the practice, the invocation name should be simple and easy as well as the same as the title of skill.

2. **Intent:** Before designing the frontend and backend of a voice user interface, it is good practice to think of the features or behaviors a skill should have. These behaviors are the intents of the skill. A skill can exhibit a set of intents that provide different actions that a user can do with the skill. Each intent defines a specific behavior. These intents determine the functionality of a skill.

3. **Utterances:** Utterances are the phrases and words that map user requests to correct intent. The purpose of utterances is to invoke intents. Utterances are mapped to skill intents and therefore form the interaction model.

   For practice, designer needs to think about speaking of user for input, which kind of phrases user might say to which skill should be able to respond with appropriate action. Alexa obtains the textual response from user command and matches it with the utterances from a set of utterances of different intents. If the phrase matched with available utterance in a specific intent repository, Alexa invokes the specific intent and then intent process the request.

4. **Slot type:** Slot type are defined by the set of variables for intent. Generally, Alexa provides built in slot types like, numbers, literals, countries, cities and so on. However, slot type are the variables that meant to define by users. From custom slot type, the designer can define any variable that will be used to identify intent.

5. **Configuration:** Configuration is responsible to integrate the interaction model with hosted service. AWS has provided all the required configurations in a single tab of a developer console. It exhibits Transport Layer Security (TSL) certificate which requires by endpoint for deployment to the web server. TSL develop communication between skill intents and cloud service.

   In addition to these customized components, there is one default component which is wake word 'Alexa'. It wakes the device and notify it that user wants to interact. 'Alexa' is a wake word for all voice enabled devices. Following the wake word, users must use a starting phrase like ask, tell etc. as referred in 'Alexa developer documentation'.

From the above example, the components of custom skill are:

Wake word: Alexa

Invocation name: History master

Intent: {Historicaldata}, this set will have variable define by slot type.

Slot type: {specificdate |14 August, 6 December, 25 December, ...}, slot type has different values which will be used as conditions in function. For example, if user says, "What happened on 14 August" then with function code will execute the

commands related to '14 august' condition the same as with the other slot values.

Utterance What happened on {Historicaldata}, A developer can think of more sentences that user can speak to extract information from Alexa. For example, "History master, what historical events took place on 14 august", "History master, tell me important events took place on 14 august", "Alexa, ask History master to tell me the importance of 14 august in a calendar year".

There can be more utterances, but important elements are 'history master' which is invocation name and '14 august' which is slot value defined in {Historicaldata} intent.

Dialog Model:

Sometimes, conversation between Alexa and user can take multiple turns, in which Alexa asks questions and user replies to these questions. The intention of this multiple turn conversation is to acquire required slot value from user if it is undefined in the user utterance. The most common cause may be the unclearness of speech or user mentions more than one slot variables in the utterance. The dialog model is important for a process of user request. Because sometime, Alexa fails to identify the slot variable from user utterance. Conversation between Alexa and user continues until all required slots received and confirmed. In addition, the dialog can also confirm or validate slot values [29].

The questions session from Alexa is of four types:

1. **Slot Elicitation:** Alexa asks for slot values. The user then responds with either utterance that includes slot value or specific slot value.
2. **Slot Confirmation:** Alexa confirms the value of slot variable that either slot value provided by user is correct or not. The user reply is generally either in yes or no.
3. **Intent Confirmation:** Alexa verifies all the information provided by user for intent is either before invocation of intent. In answer, user replies with yes or no.
4. **Slot Validation:** Alexa validate the user provided slot values with predefined values for slot variable and prompt if validation fails and ask for correct value.

There are three ways of modeling dialog model:

1. Prompts are predefined in dialog model and Alexa use the prompts by returning Dialog.Delegate directive.
2. Each step of dialog is controlled by using Dialog.ConfirmSlot, Dialog.ConfirmIntent and Dialog.ElicitSlot.
3. Use both above ways with option of taking control when requires.

In order to understand the dialog model, consider another example in which user ask Alexa to plan a trip using skill ***"Trip Planner"***. The utterance from user can

be

User: Hey Alexa, tell Trip Planner that I want to visit Budapest.

```
Alexa sends the skill Trip_plan intent with:

dialogState: STARTED
origin: null
destination: Budapest
travelDate: null
confirmationStatus: NONE
```

Alexa: Do you want to start your journey from Helsinki, right? (Slot confirmation for 'origin' slot)
User: Yes

```
Now Alexa sends the skill Trip_plan intent with:

dialogState: IN_PROGRESS
origin: Helsinki (now with confirmationStatus: CONFIRMED)
destination: Budapest
travelDate: null
confirmationStatus: NONE
```

Alexa: When you want to travel? (Slot Elicitation for 'TravelDate' slot)
User: on 20 November 2017
Alexa: This date has passed; kindly tell me future date for your trip. (Slot Validation for 'TravelDate' slot)
User: Sorry, I meant I want to travel on 20 November 2018

```
Now Alexa sends the skill Trip_plan intent with:

dialogState: IN_PROGRESS
origin: Helsinki (now with confirmationStatus: CONFIRMED)
destination: Budapest
travelDate: 20-11-2018
confirmationStatus: NONE
```

Alexa: I am reserving your trip to Budapest from Helsinki on November 20, 2018, is that ok? (Intent Confirmation)
User: Yes, thank you

At this stage, all information required by Skill is collected for reservation of trip. Now the dialogState is COMPLETED.

```
Now Alexa sends the skill Trip_plan intent with:

dialogState: COMPLETED
origin: Helsinki (confirmationStatus: CONFIRMED)
destination: Budapest
travelDate: 20-11-2018
confirmationStatus: CONFIRMED
```

However, if user replies with 'NO' at intent confirmation phase then Alexa will not re prompt because dialog considers COMPLETE since all required information received but intent.confirmationStatus is DENIED.

```
Now Alexa sends the skill Trip_plan intent with:

dialogState: COMPLETED
origin: Helsinki (confirmationStatus: CONFIRMED)
destination: Budapest
travelDate: 20-11-2018
confirmationStatus: DENIED
```

3.2 Cloud Based Service

A cloud service is required to process intent request. This service must be internet accessible. Skill connects to the endpoint of the cloud service, which contains code. Either Amazon Web Services (AWS) Lambda functions or Hyper Text Transfer Protocol Secure (HTTPS) can host Alexa skill. This endpoint is responsible for the action to the intent request. AWS Lambda is a server less computer service from Amazon Web Services (AWS) that runs the code in response to the event from Alexa request. Lambda function receives user requests from Alexa, which is then inspected by the code, resides in the lambda function and send response back to Alexa. It then takes appropriate action. Lambda function receives the request from Alexa in JSON format therefore, lambda must also respond to Alexa in JSON format. Lambda function can be written in Java, Python, Node.js and C#. Amazon Resource Name is a unique identifier, which is the address of lambda function. Alexa sends the request to the identifier of lambda function [30].

HTTPS Service can also do the response to the Alexa request. Thereby, any cloud provider can host the HTTPS script. In this case, Alexa sends request to endpoint of HTTPS service and in return, this service takes appropriate action and sends JSON response back to Alexa. HTTPS service can be written in any programming language.

AWS Lambda excludes the requirement of the hosting of code using any cloud-based service for Alexa custom skill [31]. Additionally, AWS Lambda runs the code only when it requires. Developer just needs to upload code to lambda function and

lambda is then responsible for executing a response to Alexa and managing other web resources. Thereby AWS Lambda excludes the complexity behind setting up hosting and managing it. Additionally, it does not require Secure Sockets Layer (SLL) certificate as well as verification of requests as permissions within AWS controls the access to function execution. However, Alexa uses Transport Layer Security (TLS) for the security of communication with AWS Lambda. The most important benefit of using AWS Lambda instead of HTTPS service is its free tier offer, which gives first one million requests per month free. Therefore, AWS Lambda used in the development of this thesis.

3.3 Authorization grant for custom skill

Alexa uses OAuth2.0 protocol for the authorization grant. OAuth2.0 allow third-party app to get access to HTTP service. The enabling can be done by either approval agreement between HTTP service and resource owner or by granting access to third-party app on its own behalf.

OAuth2.0 provides the security and privacy aspect to the Alexa skills and its customers. For a customer to use a skill, it has to link his/her Amazon account with skill. In this way, Alexa authenticates the third party app and grants a unique access token that identifies the customer. In return, skill gets the permission to access the user information required by from third-party app to perform action. All Alexa models e.g a custom model, smart home API model, music API model and video API model supports account linking feature. However, custom skill can provide functionality both by either with authentication or without authentication depending upon the requirement of a skill. Authentication can be done by either implicit grant or authorization grant.

Account linking can be done from Alexa mobile/web app. Account linking is the ability to match user's identity to the user's profile in another system. The skill must include access token and authorization URL options for account linking. Alexa mobile app uses authorization URL for user signing in a third-part app website which authenticates the user. After successful authentication, access token URL grants token which is required by skill to request for required information for intent request. Access token URL also grants a token in case or expiration of a previous token.

OAuth2.0 has defined four main roles for authorization of token:

1. **Resource owner:** The Alexa user who can allow skill to access his data available in resource server.
2. **Resource server:** The server that contains users' profiles. The resource server may or may not be owns by third-party app.
3. **Client:** Client is the skill that request for information from resource server by using access token.

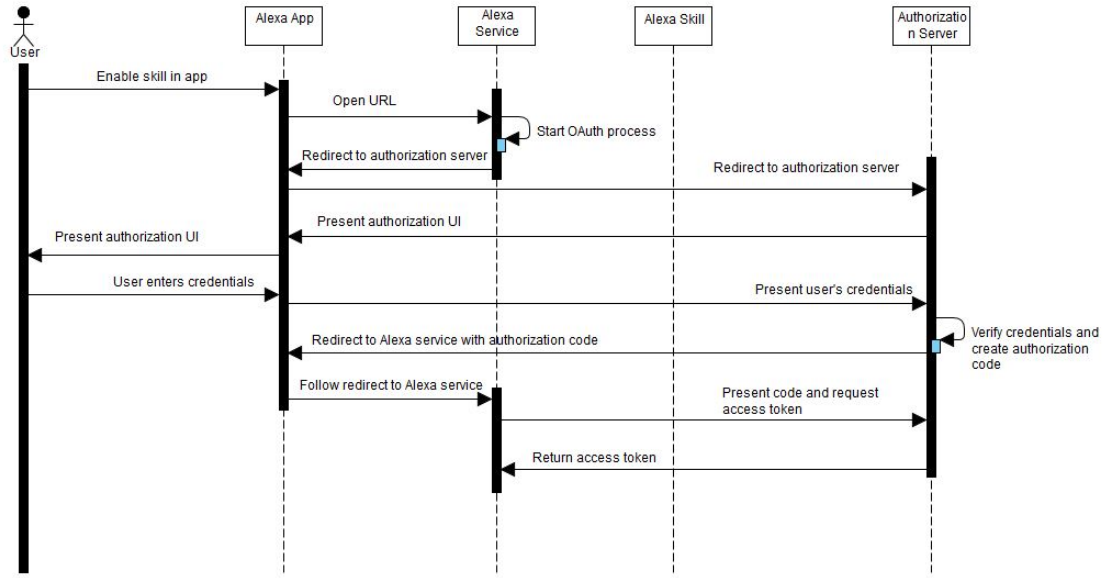


Figure 2: Skill interaction sequence

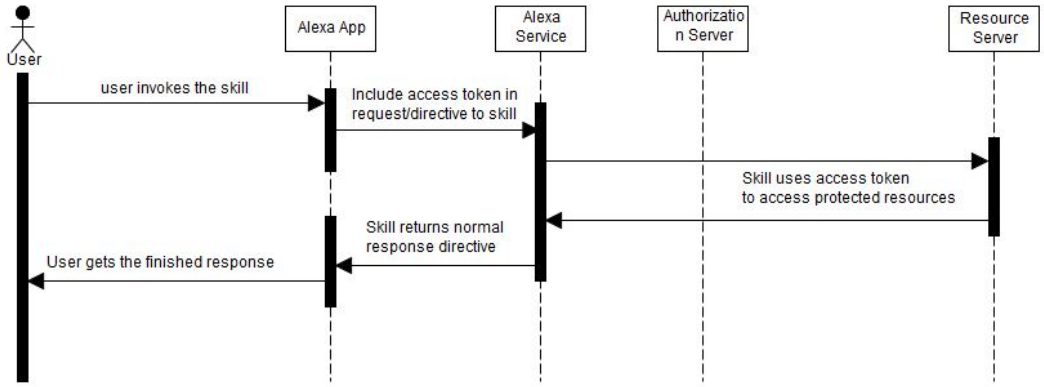


Figure 3: Authorization code grant flow

4. **Authorization Server:** The server that verifies the resource owner (user) and issues access token. This token then permit skill to retrieve required information from user's profile. Authorization server may or may not the same as resource server.

As previously discussed, access token is granted for both account linking and for information retrieval. The whole process of authorization code grant is shown in Figure 2, 3.

3.4 Alexa for Business

Amazon Alexa has provided a specialized console for organizations to use Alexa in their workplace, this is known as Alexa for Business (A4B). This console has all the required tools to map organizational matters like enrollment of users, assignment of skills to the user, create multiple rooms, assignment of group skills to each room. Other than skills available in the Alexa store, it also gives opportunity to organization to develop their own private skills. Other than this, it also enables the deployment of AVS devices and manage them as shared devices within management flow. These features make it easy to introduce voice assistant in an organization.

Private skills are those skills that are not available in Alexa skill store and only available for limited customers. The structure of private skill is the same as the custom skill. However, the only difference is to limit audience by providing ARN of AWS A4B of specific organization that intends to publish for its employees.

In order to develop the dashboard, the developer required to create AWS identity and Access Management (IAM) user with administrative permissions. After IAM permission, the developer account will act as master account. This account is capable of setting up shared AVS devices. Shared AVS devices can be either Amazon Echo, Echo Dot or Echo Plus. The A4B provides follwoing features.

1. **Manage Rooms:** In A4B, a room is a physical location in a building where Alexa devices can be placed. For example, meeting rooms, lobbies, office room.
2. **Manage Room Profiles:** Every room can have a different profile and their devices allowed for different skill to execution. The profile can differ in terms of weather, available skills, time etc.
3. **Manage Devices:** Administrator can assign different Echo devices to different rooms that can operate with their specific room profiles. However, every user can have their personalized echo with permission from the administrator.
4. **Manage Skills:** A4B has access to all available skills in the Alexa store. However, there is a provision of including private skill that is only available to particular organization. The private skill available to only those users who enrolled with the specific account of organization.
5. **Manage Skills Groups:** Administrator have the privilege to make groups of skills, it is than easy for the administrator to manage skills in different skills. For example, a skill group can be comprised of all skills related Human Resource (HR) office and then it is easy to assign all skills to new employee in HR department.
6. **Manage Users:** All users within an organization can connect their Alexa account with their organization. The administrator sends an invitation to the new user with suggestion of group of skills related to their job. When user join the organization, user can discover and enable all private skill that administrator made available to particular user's companion app. Users can

also enable those skills which are available on shared devices. User can also join the meeting on shared Echo devices using conferencing provider available in A4B console.

Additionally, A4B console also provide facility of conferencing, linking of calendar and calls but these features are only available in US region. A4B is not free tier and organization has to pay monthly for the A4B services. A4B service costs $7 per month for each shared device and $3 per month for each customer.

3.5 AWS Internet of Things(IOT)

AWS Internet of Things (IOT) is a cloud platform under Platform as a Service (PaaS) category, managed by AWS used for connecting devices easily with AWS Cloud. AWS IOT is a secured platform that provides bi-directional communication among IOT connected devices. AWS IOT has the ability to support almost billion devices and can process and securely route trillion messages to the internet connected devices and AWS endpoints.

These IOT devices can be actuators, sensors, smart appliances or embedded controllers. AWS IOT is the ability to connect other third-party cloud applications or AWS cloud applications like AWS Lambda, Amazon S3 and Amazon Dynamo DB etc. AWS IOT provides all time tracking of and communication with all devices either connected or not connected with the internet. It enables control to devices using AWS IOT Console or with the application connected with AWS IOT.

AWS IOT provides tools and the framework for the developer whereas the developer only needs to create application that can interact with devices and to collect the filtered data from devices that can be stored and used in action taking process based on certain custom rules defined in AWS IOT. The components of AWS IOT is presented in Figure 4.

Components of AWS IOT:

- **AWS IOT Device SDK:** The tool that enables the device or connectivity, authentication and communication with AWS IOT is AWS IOT Device SDK.

  In AWS IOT, all devices and applications are called Things. From the above figure, process of communication between AWS IOT and the things consists of various components. First, these devices and applications have to act as Things and access AWS IOT that can be done by using AWS IOT Device SDK that is specifically used to connect devices like Raspberry PI, Arduino etc or AWS SDK for connecting Android or other browser applications.

- **Registry:** After setting up the required tools for the devices or applications, there should be something to which devices or applications relate to at the AWS IOT console. For this, things need to be created and register on the

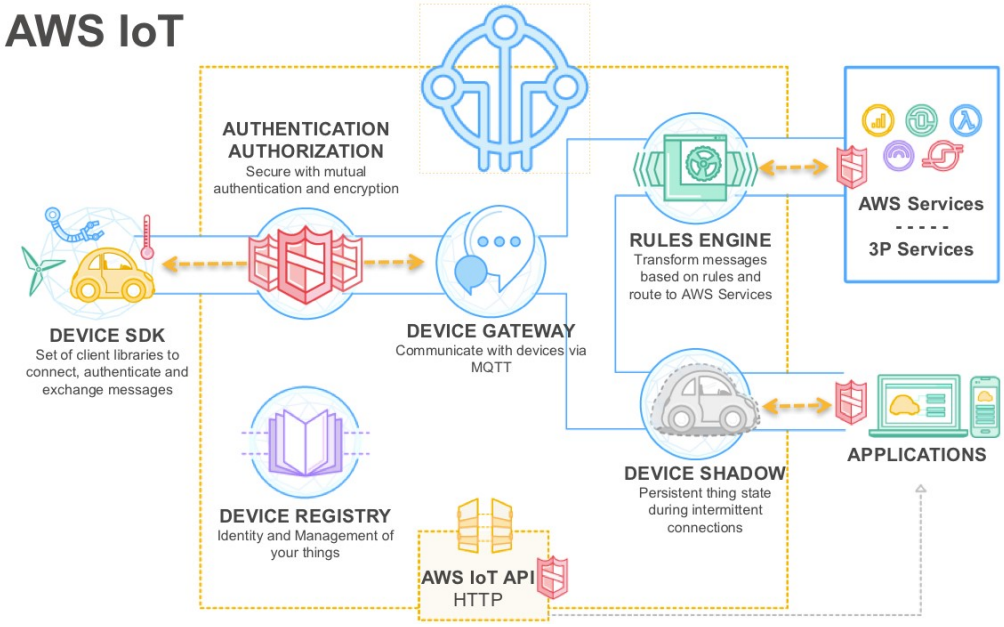


Figure 4: Components of AWS IOT

Registry of AWS IOT, which will allow keeping a record of all the devices and applications that are connected to the AWS IOT account.

The registry gives a unique identity to each device and supports metadata related to each device like capabilities and attributes of a device.

- **Device Gateway:** After the setup on both AWS IOT console and the device, communication needs to be establish between device and AWS IOT, for which the service offers three protocols for communication: MQTT, HTTP and MQTT over WebSocket.
- **Authentication and Authorization:** It is mandatory that all these connections be authenticated in order to provide a secure and safe service. This authentication varies according to the type of protocol that we are using to connect the Thing with the AWS IOT service and all the communication within the AWS IOT are using JSON data.

  MQTT protocol stands for Message Queuing Telemetry Transport, is a light messaging protocol and works around the 'publish-subscribe' model. The AWS IOT message broker connects via the port 8883, which is registered for using MQTT over SSL. The message broker or the device gateway of the AWS IOT does not inherit all the features of a standard MQTT. It does not support publishing and subscribing at QoS 2 and will not send PUBACK and SUBACK when requested for QoS 2. It does not have the retain message ability like the other MQTT brokers have, that is when a publisher requests the MQTT broker to retain the message published in a topic and send it to all the future subscribers who subscribe to the topic. If a request is made to retain messages

in AWS IOT Message broker, the connection will be disconnected.

For authenticating this protocol, which is client certificate authentication is used, which uses X.509 certificates, which are digital certificated that use the X.509 public key infrastructure standard to associate a public key with an identity contained in a certificate. X.509 certificates are issued by a trusted entity called certification authority.

HTTP protocol stands for HyperText transfer protocol, which can be used using REST API endpoint. The device gateway or the message broker allow the clients to connect via this protocol to publish using POST method. This protocol can be used only for publish to AWS IOT and cannot be used to get data back from the AWS IOT. For authentication with HTTP protocol, there are two methods; one is the client certificates, which is discussed above however port 8443 is used instead of 8883 for HTTP. Other method is Signature Version 4 signing, which is the process to add authentication information to AWS request and the authentication information must be signed with an access key. Access key consists of access key ID and secret access key. To use this protocol with authentication, port 443 must be used.

MQTT over WebSockets is similar to MQTT; the only difference is that the Websockets connection forms an outer pipe for the MQTT protocol. A WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The message broker places the MQTT packet on a WebSocket packet and sends it to the client. The client then unpacks the MQTT packet from the WebSockets packets and then processes it as a normal MQTT packet. This is useful because WebSockets allows us to receive MQTT data directly into a web browser and this protocol requires authentication through the Signatures Version 4 signing and port 443 must be used.

- **Rules Engine:** AWS IOT provides the rules engine, which easily integrates the AWS services to the devices connected and triggers various services based on data received. It accesses the incoming messages published in AWS IOT, then transform and deliver these messages to a cloud with defined rules or another thing. Rule can be applied to data coming from many devices can actions can be taken in parallel.
- **Device Shadow:** AWS IOT provides a feature called Device Shadow or Thing shadow, which is a JSON document that is used to store and retrieve current state information for a thing. A thing shadow is maintained for each thing that is connected to AWS IOT. Each thing shadow is uniquely identified by its name. It uses MQTT topic to facilitate communication between application and devices.

4 System Design

As described in Section 1.2, the two main goals of this project are:

1. Select best possible user building interaction type based on comparative analysis
2. Design and implement UI prototype that can integrate with existing BAS using cloud to cloud communication

From the comparative analysis in a Section 2.4, Alexa platform was selected for UI development, which uses the voice as user building interaction. The comparative analysis was mainly based on flexibility, scalability, security, portability and accessibility. As discussed in a chapter 3, Infrastructure build on AWS is very easily scalable and its integration with Alexa Skills kit is seamless. To achieve second goal, a UI prototype is implemented to evaluate its technical feasibility.

This chapter describes the details about the design concept, functional requirement and system architecture that will lead towards development of UI prototype for an office building. Moreover, the Chapter 5 will explain in detail the implementation and deployment process of the system.

4.1 Functional Requirement

The main idea is to implement UI in the office building where the users will be the employees working in the building. The responsibility of UI must be able to assist users in executing nonproductive tasks to save time for the users. The nonproductive tasks can be sending a leave email, turning on/off any electrical appliance, weather updates, news feed and conferencing. In addition to this, there will be multiple users and multiple rooms in the building. Therefore, a designed system must be dynamic enough to accommodate multiple users and rooms profile.

The functional requirement of the system require to posses following capabilities:

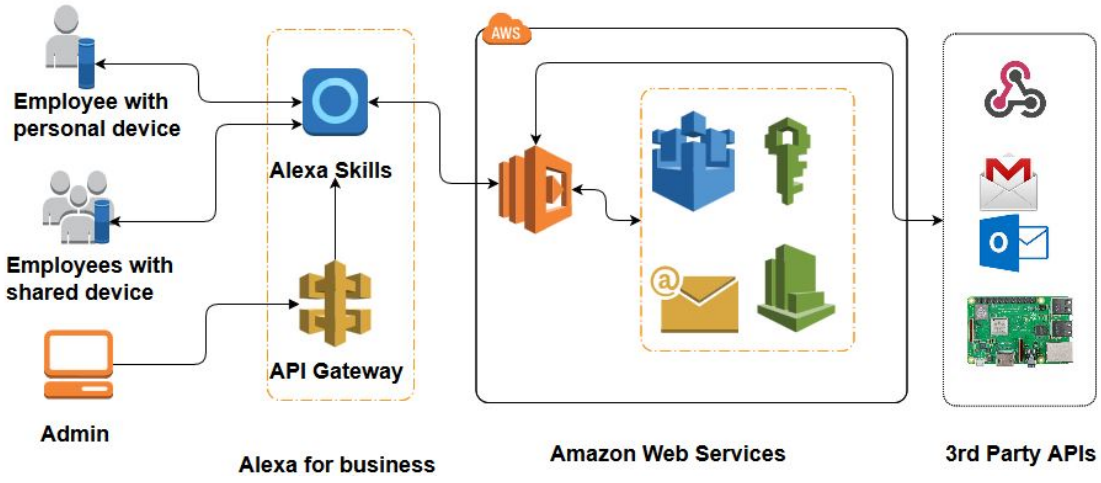


Figure 5: Overview of system architecture

- Control electrical appliances such as bulb, fans, display screen and many others.
- Execute various tasks such as writing a leave mail, reporting IT Maintenance and many others.
- Enroll multiple users and provide flexibility to all users.
- Provide security and privacy through login authentication.
- Capability of creating different room profiles.

In view of above discussion, a concept is designed for UI prototype, which titled as **'Alexa Office'**. It must have the ability to integrate with the existing system and exhibit following skills:

1. Smart home skill
2. Visitor skill
3. Mail skill
4. IT Maintenance skill

4.2 System Architecture

The overview of system architecture is ilustrated in Figure 5. The system has five main components:

1. Alexa device
2. Alexa skills
3. Alexa for Business

4. Amazon Web Services
5. Third party APIs

The user interacts with the Alexa skills through an Amazon Alexa device. It could be either Amazon Echo home speaker or Alexa companion app. The device sends task request to the Alexa Skill which lies in Alexa cloud. Alexa Skill extracts the intents and slot values from the request and pass it in the form of JSON request to the corresponding AWS Lambda function that resides in the AWS Cloud. AWS Lambda function serves as backend service to the Alexa skill as explained in Section 3.2. Whereas Alexa skill is created using Custom skills model. Custom skills are discussed in section 3.1.

AWS Lambda function analyzes the request and send commands to respective service for execution, for example, to change the power status of electrical device, it sends the message to AWS IOT to change the power status of a device. The services can be either belongs to the AWS family or third party APIs depending upon the request. After execution, service returns the successful message to the AWS Lambda function which in return generate a JSON response and send back to Alexa where the generated message is presented to the user.

The backend service handles HTTPS requests, just like a web backend application. In that sense, the architecture is similar to web app. In order to create a maintaianable and easily extendable application. The AWS Lambda function is structured to handle three main request types and these are the launch, intent and session request. In case of launch or session end request, either a response welcoming the user to the skill or a goodbye response is generated. However, if the incoming request is of intent type, then response generation is delegated to the respective services related to the invoked skill.

Each skill requires some background knowledge in order to integrate services either AWS or third-party APIs with Alexa Skills. Therefore, background study is conducted to gain significant knowledge of tools other than described previously, which is useful for development of specific skill. These tools are mostly belongs to AWS ecosystem such as use of Cloud watch for the debugging of lambda function. Cloud Watch is an AWS utility that registers every event from AWS Lambda [32].

5 Implementation

As described in Section 4.1, there are four skills that needed to be developed and implemented in **'Alexa Office'**. This chapter presents the development of all the skills. Each skill requires development of Interaction Model, AWS Lambda function and integration of required services with AWS Lambda function. Morever, this chapter also introduces tools and technologies that used to develop each skill. The Section explains the complete implementation of all skills.

5.1 Setup of AWS Account

The AWS account needed to be set before the development phase. All skills require account linking for enablement as discussed in 3.3. Therefore, the developer account should be authenticated before the developement of Alexa skills. OAuth2.0 provider is required to implement this process. The steps involved in authenticating account with OAuth2.0 are:

1. Authentication of AWS account through log in Alexa developer console
2. A new security profile is created.
3. Client ID and Client Secret key is obtained from web settings.

In order to develop these skills, the region was set at US east region. The reason behind choosing US east region is that Alexa for business is only available in US east region and AWS has been launched in Europe region in July 2018.

5.2 Room Lamp Skill

The room lamp skill is developed to test the functionality for control of devices using Alexa. To successfully implement the skill, hardware is developed on Raspberry Pi3 and Relay. With this architecture, any device can be controlled by Alexa. In this

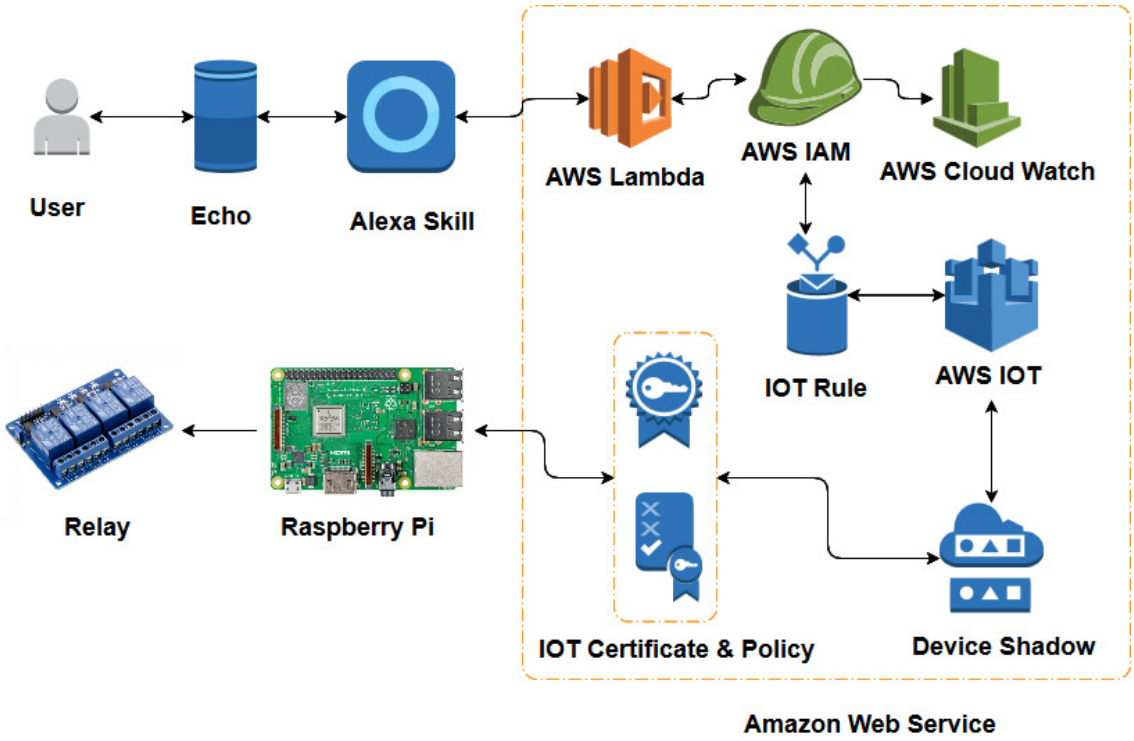


Figure 6: Architecture of *Room Lamp Skill*

case, a room lamp is taken as an external device. The complete overview of this system is shown in figure 6.

The important task is to enable cloud to cloud communication because Alexa can only control any device through its device cloud. AWS IOT platform is capable of establishing device cloud as discussed in 3.5 therefore, AWS IOT used for device cloud that can store and update device status. The voice command sent to the AWS Lambda from Alexa device in JSON request format. AWS Lambda function named *LampSkill* served as endpoint for the mentioned skill. AWS Lambda function processes the request and sent the control commands to AWS IOT Thing. The communication between lambda function and thing is based on defined rules in the rule engine as discussed in section 3.5. The Thing then update the status of device shadow.

The development of *Room Lamp Skill* required both hardware and software implementation. In hardware implementation, a device has connected with AWS IOT based device cloud. The process consists of the following steps.

1. Setup of Device
2. Configuration of AWS IOT
3. Setting up communication between Raspberry Pi3 and AWS IOT

Whereas software implementation consist of:

1. Creation of Interaction Model
2. Creation of AWS Lambda function
3. Setting up communication between AWS Lambda function and AWS IOT

Setup of Device:

The components used for setting up a device are Raspberry Pi3 and Relay. The Raspberry Pi3 is a minicomputer with the capability of controlling the entire smart home system, which is capable of processing programming codes and controlling a huge number of devices via its GPIO pins. For specifications, the Raspberry Pi3 processes at 1.2GHz with a 64-bit processor, exhibiting a performance of around 10x that of original Raspberry Pi. It has 40 GPIO pins, 4 USB ports, a full size HDMI port, micro-SD port and powered by a micro USB power source. For wireless communication, the Raspberry Pi3 has both WiFi and Bluetooth module on board.

For this project, Raspberry Pi is connected with the computer via the Secure Shell (SSH) and PuTTY which is one of the SSH free client is used for Windows platform. A script is written to control the Relay using GPIO pins.

Configuration of AWS IOT:

To connect a device with AWS IOT, a thing is created that represents the physical device and its certificates are issued to register the device with AWS IOT. The configuration of AWS IOT required following two steps:

1. **Register a Thing:** A thing named *"Lamp"* is created that represents a physical lamp whose status or data is stored in the AWS Cloud. The lamp's status or data is stored in a JSON document known as device's shadow. The device shadow is used to both store and retrieve state information.
2. **Attach certificates and policy with the thing:** SSL Certificates are created for the device which is stored in Raspberry Pi3. These certificates are used to authenticate the requests and therefore must remain active while the device is in use. These certificates are public key, private key and the root CA Certificate which is X.509 certificate.

   A policy named *"RPI Policy"* is attached with the thing. Policy is used to check whether the requesting resources are authorized or not. The policy for 'Lamp' thing is shown in Listing 1.

```
1
2  {
3    "Version": "2012-10-17",
4    "Statement": [
5      {
6        "Effect": "Allow",
7        "Action": "iot:*",
8        "Resource": "*"
9      }
```

```
10      ]
11    }
```

Listing 1: RPI Policy

Without certificates and policy, AWS IOT does not allow any software to publish or subscribe the MQTT messages from/to topic.

Communication between Raspberry Pi3 and AWS IOT:

The communication between Raspberry Pi3 and AWS IOT is based on MQTT protocol. Therefore, 'AWSIOTPythonSDK' tool is installed in Raspberry Pi3 and 'AWSIoTMQTTClient' is added to device's script which is imported from 'AWSIoT-PythonSDK.MQTTLib'. The script used to connect the physical device with AWS IOT is given in Listing 2. The host represents endpoint AWS IOT device shadow and port 8883 used for MQTT protocol.

```
1
2
3  host = "xxxxxxxxxxxxxx.iot.us-east-1.amazonaws.com"
4  rootCAPath = "/home/pi/certs/rootCA.pem"
5  certificatePath = "/home/pi/certs/certificate.pem.crt"
6  privateKeyPath = "/home/pi/certs/private.pem.key"
7  clientId = "Lamp"
8  topic = "$aws/things/Lamp/shadow/update/accepted"
9
10 myAWSIoTMQTTClient = None
11 myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId)
12 myAWSIoTMQTTClient.configureEndpoint(host, 8883)
13 myAWSIoTMQTTClient.configureCredentials(rootCAPath, ...
       privateKeyPath, certificatePath)
```

Listing 2: Enabling of MQTT Protocol between device and AWS IOT

Interaction Model:

As discussed in section 3.1, interaction model maps the interaction between the user and Alexa. Custom skill model is used for this purpose. A JSON schema is created that contained an invocation name, intents, slot values and sample utterances are shown in Listing 3.

```
1
2   {
3     "interactionModel": {
4       "languageModel": {
5         "invocationName": "room lamp skill",
6         "intents": [
7         {
8           "name": "AMAZON.FallbackIntent",
9           "samples": []
10        },
```

```
        {
          "name": "AMAZON.CancelIntent",
          "samples": []
        },
        {
          "name": "AMAZON.HelpIntent",
          "samples": []
        },
        {
          "name": "AMAZON.StopIntent",
          "samples": []
        },
        {
          "name": "AMAZON.NavigateHomeIntent",
          "samples": []
        },
        {
          "name": "PowerIntent",
          "slots": [
          {
            "name": "powerResult",
            "type": "power_mode"
          }
          ],
          "samples": [
          "Kindly turn {powerResult} the room lamp",
          "Turn {powerResult} on the lamp",
          "Turn {powerResult}"
          ]
        }
        ],
        "types": [
        {
          "name": "power_mode",
          "values": [
          {
            "name": {
              "value": "off"
            }
          },
          {
            "name": {
              "value": "on"
            }
          }
          ]
        }
        ]
      }
    }
  }
```

Listing 3: JSON interaction schema for Room Lamp Skill

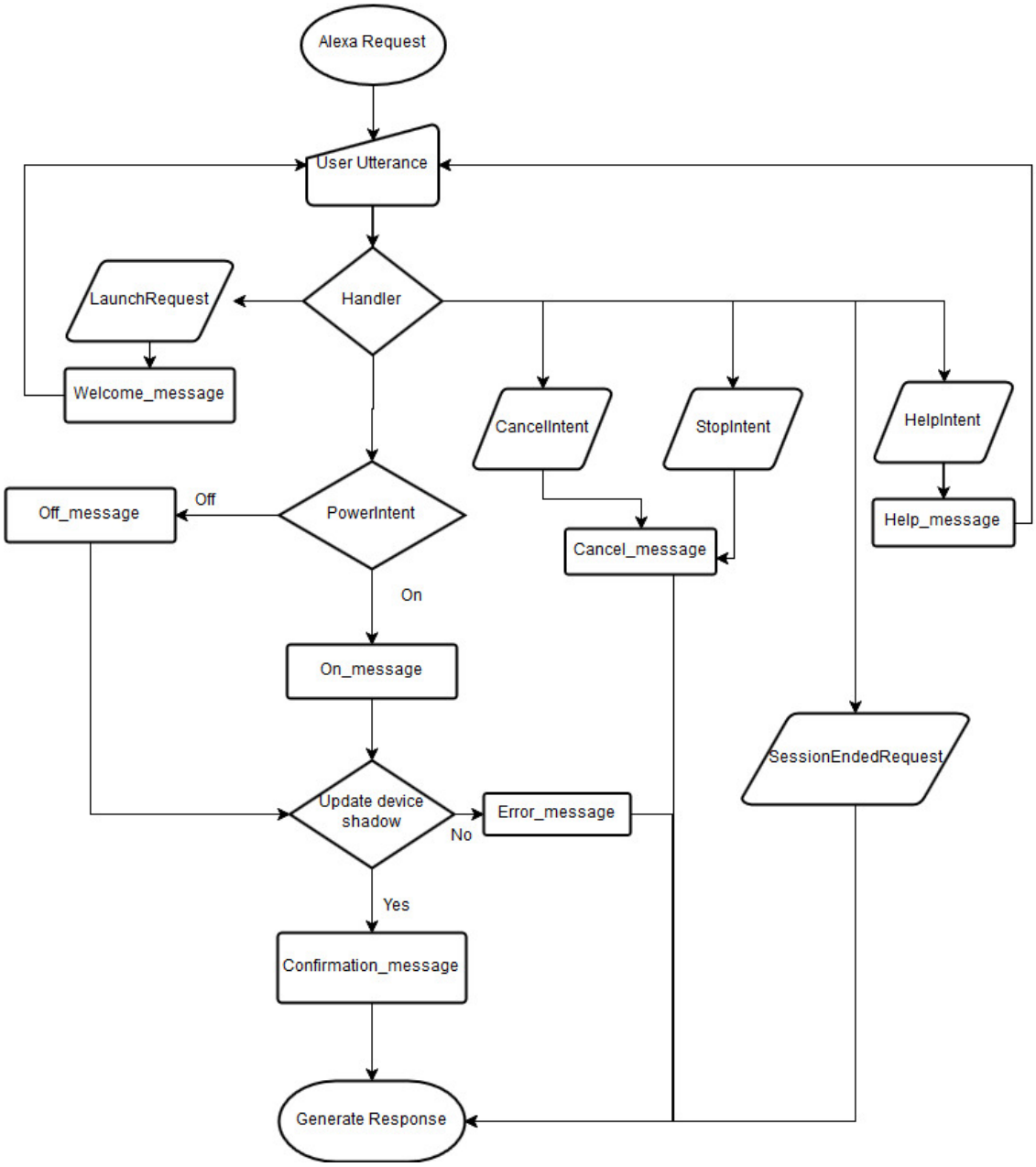


Figure 7: Control flow diagram of *Room Lamp Skill*

A custom intent *'PowerIntent'* is created for two values that are on and off from user utterance. Alexa collects the required information from user utterance and parsed into JSON request that is sent to the AWS Lambda function. A sample JSON request of this skill is shown in Listing 4.

```
1
2 "request": {
3     "type": "IntentRequest",
4     "requestId": ...
```

```
        "amzn1.echo-api.request.93487c6c-9159-4e6a-ae11-cd602ec76dfa",
    "timestamp": "2018-12-02T09:05:07Z",
    "locale": "en-US",
    "intent": {
      "name": "PowerIntent",
      "confirmationStatus": "NONE",
      "slots": {
        "powerResult": {
          "name": "powerResult",
          "value": "on",

        },
        "confirmationStatus": "NONE",
        "source": "USER"
      }
    }
  }
}
```

Listing 4: JSON request of Room Lamp Skill

AWS Lambda function:

As discussed in Section 3.2, AWS Lambda is used for the deployment of function. AWS Lambda function named **'LampSkill'** is created from the *alexa-skill-kit-sdk-factskill* template and Node.js 6.10 runtime was selected. The code written for execution of tasks related to this skill is illustrated in Figure 7.

A function named *LaunchRequest* is called whenever the invocation name of skill is invoked without providing slot values for intents. In response to *LaunchRequest*, Alexa asks for required intents for execution of tasks through welcome message. The sample conversation would be:

User: Open room lamp skill.
Alexa: Welcome to Room Power Control Dashboard. Do you want to turn your lamp on or off?

However, if user utterance already contains valid intent's slot value then defined function named *PowerIntent* is triggered. The slot values are either 'On' or 'Off' for power intent. The sample conversation could be:

User: Ask room lamp skill to turn off the lamp
Alexa: Turning off the lamp!

During execution of *PowerIntent*, *updateshadow* function is called with updated parameters of device shadow and after its execution, *updateshadow* sends either the confirmation message to device console for successful execution or error message in case of error.

In addition to the above mentioned functions, *CancelIntent* and *StopIntent* is also created. The purpose of these intents is to terminate the process. User can call either of these intents during conversation and Alexa terminates the process and

generate response with cancel message "I am cancelling everything, to start over, please invoke the skill again, Thank you". Whereas the purpose of *HelpIntent* is to guide user about the skill through message "You can turn on or turn off the lamp" but it does not end session in fact ask user for intents.

After completion of a task, AWS Lambda function generates JSON response and sent back to Alexa to inform user with a respective prompt message.

Setting up AWS Lambda with AWS IOT:

To connect AWS Lambda with AWS IOT, a special IAM role is created with permissions to create AWS IOT things and rules and lambda function. In IOT Core console, a rule is created for the triggering of lambda function in case of an event. In addition to this, an additional function is defined as '*updateshadow*' which is provided in Listing 5. The purpose of this function is to send desired value to **Lamp** device shadow in AWS IOT.

```
1
2  function updateShadow(UserData, EmitEvent) {
3    var AWS = require('aws-sdk');
4    AWS.config.region = "us-east-1";
5    var Update_Parameters = {
6      "thingName" : "Lamp",
7      "payload" : JSON.stringify(
8        { "state":
9          { "desired": UserData
10         }
11       }
12     )
13   };
14   var iotData = new AWS.IotData({endpoint: ...
       "xxxxxxxxxxx.iot.us-east-1.amazonaws.com"});
15   iotData.updateThingShadow(Update_Parameters, function(err, ...
       data)  {
16     if (err){
17       console.log(err);
18       EmitEvent();
19     }
20     else {
21       console.log("Updated thing shadow");
22       EmitEvent();
23     }
24   });
25 }
```

Listing 5: *updateshadow* function

5.3 IT Maintenance Skill

Maintenance is required everywhere in an organization and employees report them frequently through various ways either in person or sending messages to the maintenance team through some means. The idea of ***IT Maintenance Skill*** is to report IT related issues to the IT Department of the building. The message is sent through Slack channel. Slack can be the way by which maintenance team get alerts during their available hours in the organization. The architecture of IT Maintenance Skills presented in Figure 8.

Slack

Slack is a tool that brings team communication and collaboration into one place, which is widely used for internal communication in an organization.

This channel of slack workplace must receive messages from the Alexa skill, for this purpose incoming webhooks app is used. Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of HTTP normal requests with a JSON payload [33]. HTTP uses the request-response model to communicate between the client and server. There are two widely used request methods GET and POST. GET is the request method where the client sends a request to get data from the server. POST is the request method where the client sends a request to Post data into the server or endpoint. Incoming webhooks app generates an endpoint to which Alexa sends a JSON payload.

Add the configuration to the maintenance channel. This is the webhook URL, the endpoint to which JSON payload is sent using POST method. The text property in the JSON payload refers to the simple message that is to be post on the channel. Write method requires the data has to be sent as the request body to be a string. In this case, JSON.stringify method used to send JSON as a sting.

The steps involved in development of *IT Maintenance Skill* are:

1. Setup of Incoming Webhook app in Slack
2. Creation of Interaction Model
3. Creation of AWS Lambda function

Setup of Incoming Webhook app in Slack:

A new workplace is created in slack named as “Smart Building Research Group”. A new channel named as “aalto-it” is created. The username for Webhook is “Alexa”. A PostSlack function is created with the message and a callback function. The script of PostSlack function is presented in Listing 6.

```
1
2 function PostSlack(message, callback){
3   var https = require('https');
4   var url = require('url');
```

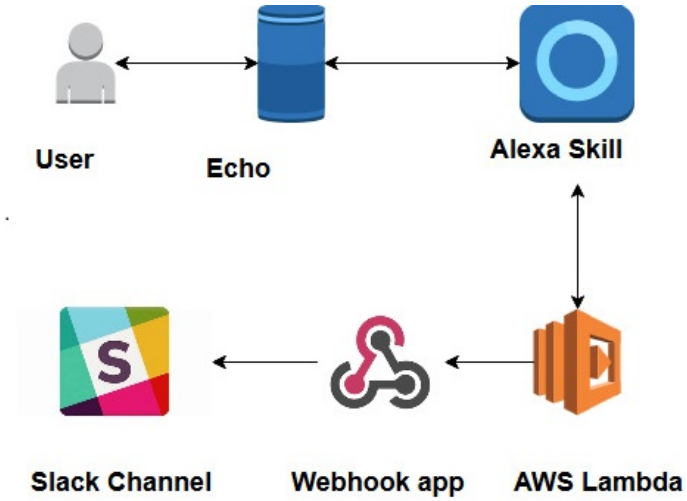


Figure 8: The architecture of IT Maintenance Skill

Table 5: Interaction Model of *IT Maintenance skill*

Invocation Name	**IT Maintenance**		
Intents	IssueIntent	DetailsIntent	
Intent Slot	issues	name	places
Slot Values	Desktop, Software, Internet, Laptop	Default Slot type is used for name identification	3555,3554

```
5   var webhook_url = 'xxxxxxxxxxxxxxxxxx';
6   var request_object = url.parse(webhook_url);
7   request_object.method = 'POST';
8   request_object.headers = {'Content-Type': 'application/json'};
9
10  var request_instance = https.request(request_object, function ...
        (response) {
11    if (response.statusCode === 200) {
12      callback();
13    } else {
14      console.log('status code: ' + response.statusCode);
15    }
16    });
17    request_instance.write(JSON.stringify({"username":"IT-Assistant", ...
          text: message}));
18    request_instance.end();
19  }
```

Listing 6: *PostSlack* function for sending message to webhook url

Creation of Interaction Model

For *IT Maintenance skill*, the custom skill model is used for this interaction design. A JSON schema is created that contained an invocation name, intents, slot values and sample utterances are given in Table 5. There are two intents included in the interaction model that are:

1. **IssueIntent:** The intent is used to get the values of issues like software related issues, internet related issues or other IT related issues.
2. **DetailsIntent:** This intent is used to collect user information like user's name and room location in building.

For this skill, Dialog model is also included to acquire valid slot values for required intents. As described in section 3.1, it is very important to attain all required information and to confirm that Alexa has understood correctly what user has uttered otherwise there is chance of sending an incomplete or wrong report to the IT department. In this case, slot elicitation, slot validation and slot confirmation is used in addition to intent confirmation. In case of slot elicitation, When user tell Alexa about the issue then Alexa will ask for the name and location for the slot filling. Alexa will validate the slot value and confirm it from the user. One such sample conversation would be:

User: I am facing issues with the laptop
Alexa: We are sorry to hear that you are facing issues with laptop, please tell us your name so that the maintenance team can address your issue. (Slot Elicitation)
If user replies with his name and Alexa failed to understand it then Alexa will reply as:
Alexa: Can you please repeat your name?(Slot Validation)
User: My name is John
Alexa: A maintenance report will be send in the name John , is that ok?(Slot Confirmation)

This purpose of this dialog model is to collect correct user's identity. In case, Alexa misunderstood it then it will become for IT department to locate the user. Therefore, the dialog model is also used in AWS Lambda function.

Creation of AWS Lambda function:

AWS Lambda function named **'ITSkill'** is created from the *alexa-skill-kit-sdk-factskill* template and Node.js 6.10 runtime was selected. The code written for execution of tasks related to this skill is illustrated in Figure 9. Request handler for this skill is the same as *Room Lamp Skill*. However the dialog model is included using *DelegateSlot* function. In addition to launch, stop,cancel and stop requests, Following intents are used to obtain required information from the user:

- **IssueIntent:** The purpose of this function to collect slot values related to issueIntent included in Interaction Model. This function generates a message response to Alexa to collect required intent values to complete the request.

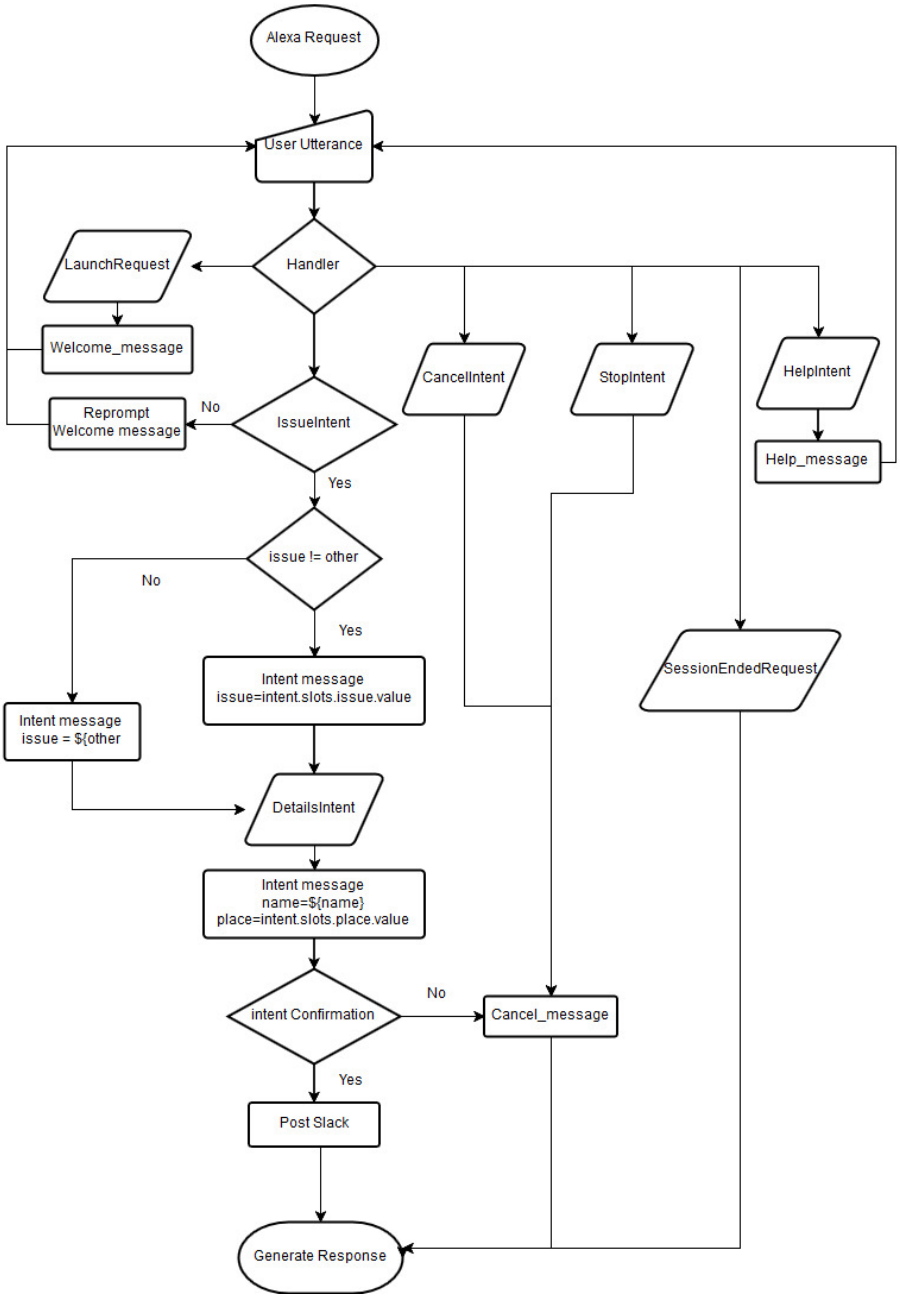


Figure 9: Control flow diagram of *IT Maintenance skill*

- **DetailsIntent:** This function stores the slot values of name and place and generates a intent confirmation response to Alexa to confirm the user's intent.
- **YesIntent:** This function utilizes the Amazon inbuilt intent 'Amazon.YesIntent' for intent confirmation. The purpose is to receive a postive response from the user before execution of the task. With 'yes' confirmation, it calls *PostSlack* function to send the report to the slack channel.

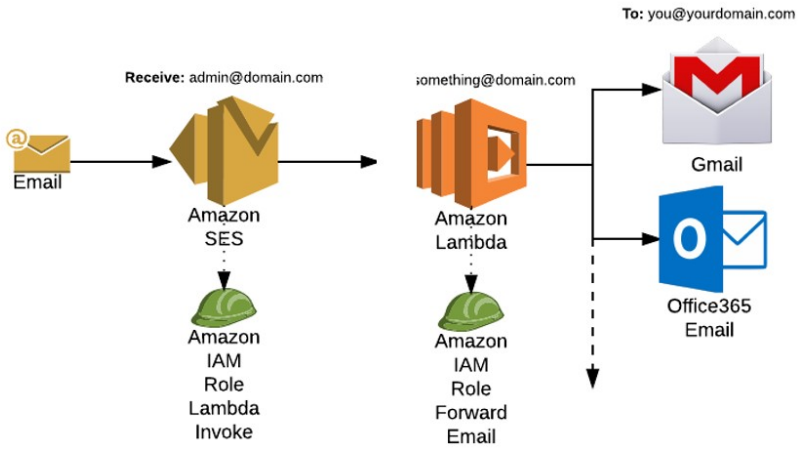


Figure 10: Architecture of *MailSkill*

5.4 Mail Skill

Mail is a formal way to communicate within the organization. It is one of the ways to have messages written down digitally. The advantage when it is written down is that it stays forever and can be referred to whenever required. The difficulty that lies while building this kind of a skill is that, specific conversations have to be build that will help the skill to gather all the details that are required for the specific use case.

The ***Mail Skill*** is used to send a mail that contains the leave details of an employee, like how long the leave is for, from the date, to date and the category under which the reason for the leave will fall. All these details have to be precise; therefore, it is good to have confirmation of every detail from the user. Thereby, not even a single detail that is being misunderstood and lost because of the skill. The architecture used for *Mail Skill* is illustrated in Figure 10.

Setup of Amazon SES:

Amazon's simple email service is an email platform that provides an easy cost effective way to send and receive emails using own email addresses and domains [34]. For this skill, SES is used to send one mail id to another mail id. The setting up of Amazaon SES includes the following steps:

SendEmail function is defined with the parameters object from the session attributes is presented in Listing 7 . eParams object is defined in the format required by send email function of the SES object. The call back function is also provided with data and error for SES object. If there is an error, then error value is logged and emits the event error message as the speech output. If the mail is sent without any error, then it emits the event with the successful message as speech output.

Table 6: Interaction Model of *Mail skill*

Invocation Name	**Mail Skill**					
Intents	LeaveIntent					
Intent Slot	Reason	Name	Recipient	days	From date	To date
Slot Values	Personal, Vacation, Medical	Amazon. US_FIRST _NAME	Amazon. US_FIRST _NAME	Amazon. NUM- BER	Amazon. DATE	Amazon. DATE

```
1
2  function sendEmail(parameters){
3    var eParams = {
4      Destination: {
5        ToAddresses: [mailId[parameters.to]]
6      },
7      Message: {
8        Body: {
9          Text: {
10           Data: `I would like to take ${parameters.days} days ...
                 leave from ${parameters.fromdate} to ...
                 ${parameters.todate} \n Reason:    ...
                 ${parameters.reason}`
11         }
12       },
13       Subject: {
14         Data: "Applying for leave"
15         }
16       },
17       Source: mailId[parameters.from]
18   };
19   ses.sendEmail(eParams, function(err, data){
20     if(err){
21       console.log(err);
22       self.emit(':tell','There was an error while requesting ...
            the mail service');
23     }
24     else {
25       self.emit(':tell',`A leave mail has been sent from ...
            ${parameters.from} to ${parameters.to}, Happy holidays`);
26     }
27   });
28 }
```

Listing 7: sendEmail function for *Mail Skill*

Interaction Model:

The interaction model for *Mail Skill* is presented in Table 6. The objective of this interaction model is to collect information related to leave application which is, the purpose of leave, duration of leave and reciever email address for this mail. The

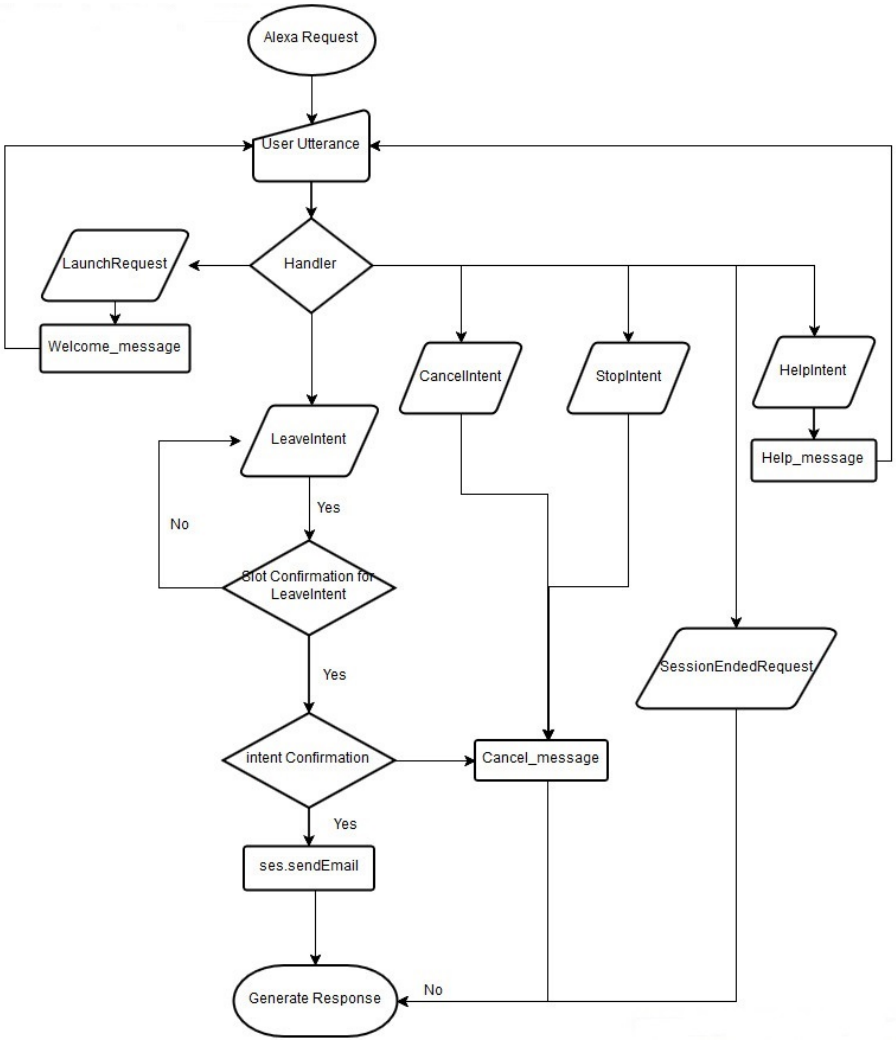


Figure 11: Control flow diagram of *MailSkill*

only intent used for this skill is:

LeaveIntent: The objective of this intent to gather leave details that include the sender, receiver, duration of leave, from the date, for date and reason for leave.

Dialog model is also used in *Mail Skill*. It includes Slot Elicitation, Slot Validation, Slot Confirmation and Intent Confirmation. A sample section of conversation for this skill would be:

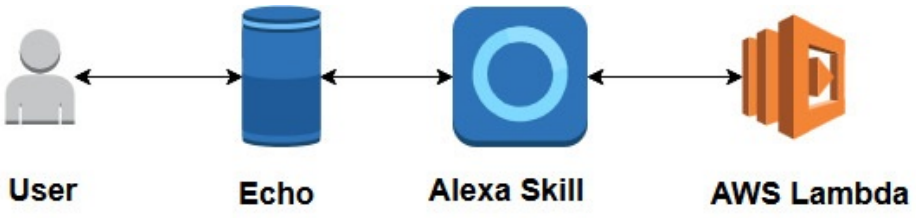


Figure 12: Architecture of *VisitorSkill*

Alexa: Medical, vacation or Personal, which will be the appropriate reason for your leave. (Slot Elicitation)
User: Medical
Alexa: Your leave is for medical reason. Am i correct?(Slot Validation)
User: Yes
Alexa: Are you sure you want to send an email to John, regarding a leave for 2 days from 2018-11-25 to 2018-11-26 for medical reasons?(Intent Confirmation)
User: Yes
Alexa: A leave mail has been sent from Alex to John, Happy holidays

AWS Lambda function:

The lambda function is named as "MailSkill". The flow chart of its script presented in Figure 11. As the purpose of this skill is to collect information and then execute it therefore, *DelegateSlot* function is used for prompt response and the dialog model in the AWS Lambda function. After successful confirmation of the *DelegateSlot* function, *ses.send* function is called that sends email to the mentioned receiver using Amazon SES service.

5.5 Visitor Skill

The idea of visitor skill is to guide and provide necessary information related to room facilities for the person who visits the room. The visitor skill greets the visitor and tells him/her about available Alexa Skills for particular room. For example, if the visitor is intended to use meeting room then visitor skill will respond to the name of skill that can help him turning on the electrical appliances and setup room for as per the visitor requires by using help of other skills. The architecture of *Visitor Skill* is illustrated in Figure 12.

Interaction Model:

This skill is relatively simple as it does not require any additional inputs to fill intent's slot value therefore only one intent is included, which is used to get the name of the visitor. The interaction model for Mail Skill is presented in Table 7.

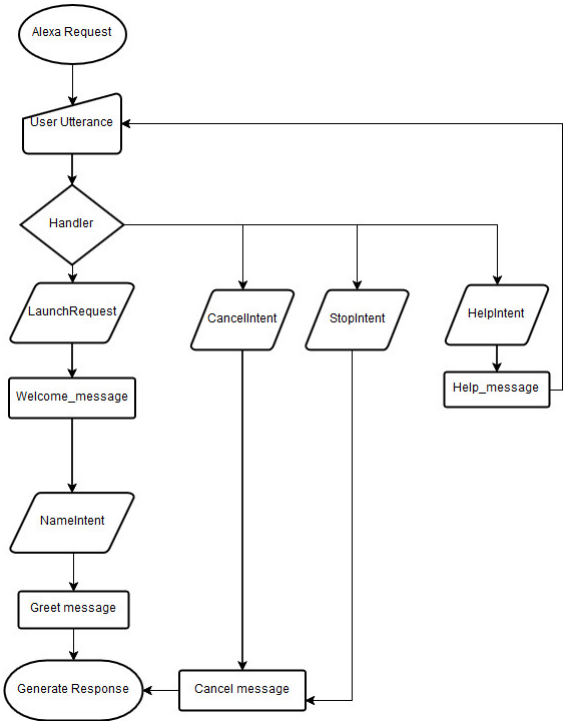


Figure 13: Control flow diagram of *VisitorSkill*

Table 7: Interaction Model of *Visitor skill*

Invocation Name	**Visitor Skill**
Intents	NameIntent
Intent Slot	Name
Slot Values	Amazon.US_FIRST_NAME

AWS Lambda function:

The lambda function is named as "VisitorSkill". The flow chart of its script presented in Figure 13.

5.6 Setup of Alexa Office

As discussed in Chapter 4, the system must be capable enough to include multiple users and rooms with their individual profiles. Alexa for Business platform is used to set up Alexa office in where developed skills will be implemented and will be made available to all the users in the building. A4B provides the template to create and manage profiles of users and rooms as discussed in Section 3.4. Setup of Alexa office requires the following steps to get functional.

1. A new IAM user is created, named as "A4B". This user created within the same account that is used for development of Alexa skills.

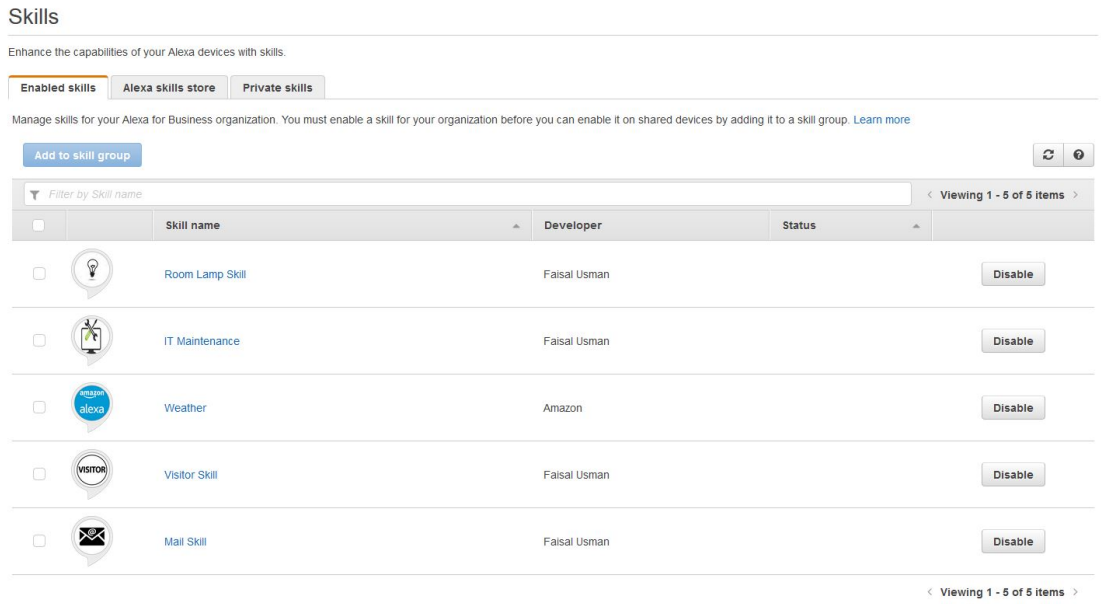


Figure 14: Available Alexa skills for Alexa Office

2. An existing policy named as "AlexaforBusinessDeviceSetup" is attached with the created IAM User. This policy is used to register devices with A4B account.

3. Access key ID and Secret access key are created for this account, which is to be used for registration of a device.

4. A shared device is connected with A4B by using the device setup tool. During device setup, device needs to be on the same network as other AWS services.

After successful registration of the device, A4B is functional. The user invitation option is used to invite new users to the A4B environment. The skills that are developed must be deployed using ARN of Alexa office. In this way, these skills deployed as 'Private Skills' and are only available to the users enrolled in this platform. However, the administrator can also enable Alexa skills from the Alexa store. All the developed skills must require to pass the functional test from Amazon Alexa Certification before deployment.After the fucntional test, skills are available in A4b dashboard as presented in Figure 14.

From the A4B dashboard, specific skills can be restricted to specific rooms or specific users by creating groups. Conferencing, Calls and Calendar is not used in this project as these are only available in US east region.

6 Testing

The real time testing is available through AWS with the deployed AWS Lambda Function. Testing of each skill can be done through three ways and these are:

1. **Amazon developer test simulator:**
   The service simulator resides in the Test section of the Amazon developer portal. Moreover, based on the text input in a sample utterance form, the service simulator provides a Service Request and Service Response, either successful or not. For this type of testing, there is no need to provide a full sentence for the service simulator to work. A simple sample utterance will trigger the test. A test event is a sample piece of JSON data that is passed into the Lambda function, to see what the result would have been, had it been called by the specific intent.

2. **AWS Lambda function test events and AWS Cloud Watch:** It is an easier way of testing the code, with more descriptive details about what events are triggered or where an error has occurred, is the Lambda function console under test event type which can be configured for appropriate event type. For every event, an event log is generated in AWS CloudWatch with window output and action summary. AWS CloudWatch also provides information related to event execution in the log output such as duration, billed duration, memory size and memory used.

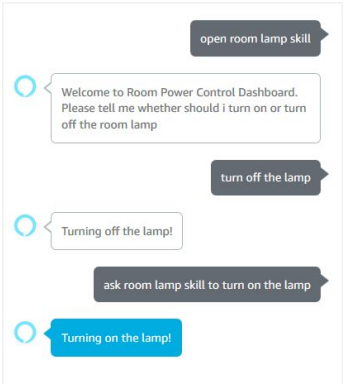
3. **Directly through Interaction with the Amazon Exho or Alexa Skill testing emulator:** During the development and testing of the skill, the Amazon Echo device was not available to use. However, this issue was resolved by an Alexa Skill testing emulator provided by "Echoism.io", that allows the developer to link their Amazon developer account with this website and test any skill assigned to their account. This type of test was the most convenient one, for the interaction between the user and the device is enclosed and the developer can construct the communication accordingly.

Each skill has gone through all these mentioned testing ways during each development phase and after deployment in 'Alexa Office'. These tests validate if a proper

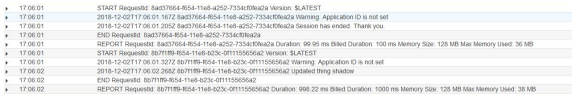
response is generated for a given request. This Chapter validates the results for the functional requirement described in Section 4.1. Sections 6.1, 6.2, 6.3 and 6.4 presents the testing results for *Room Lamp Skill*, *IT Maintenance Skill*, *Mail Skill* and *Visitor Skill* respectively. Whereas Section 6.5 discusses the result related to testing all skills.

6.1 Testing of Room Lamp Skill

The *Room Lamp Skill* includes Raspberry Pi3 and AWS IOT. Therefore, it is also tested using IOT shadow document. The testing phase of *Room Lamp Skill* is shown in Figure 15. When input is given to the Alexa developer test simulator as shown in Figure 15(a), an event is logged in AWS CloudWacth corresponding to *LampSkill* AWS Lambda function which is shown in Figure 15(b). From the log event, it can be seen thing shadow has been updated and it is verified in AWS IOT Device Shadow document as presented in Figure 15(c) and at the same time Output console of Raspberry Pi3 has acknowledged the command and executed it, which is shown in Figure 15(d).

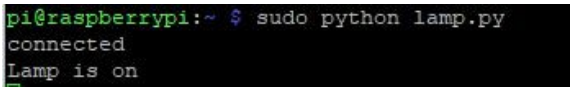


(a) Input given to the Skill using Alexa developer test simulator



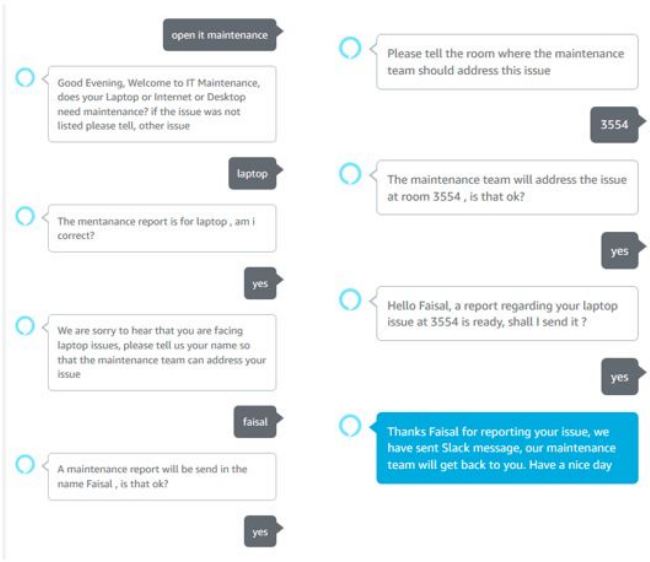
(b) Event generated for *Room Lamp Skill* in AWS CloudWatch

(c) IOT shadow docuement of *Lamp* Thing



(d) Output console of Raspberry Pi3

Figure 15: Testing of *Room Lamp Skill*



(a) Input given to the *IT Maintenance Skill* using Alexa developer test simulator

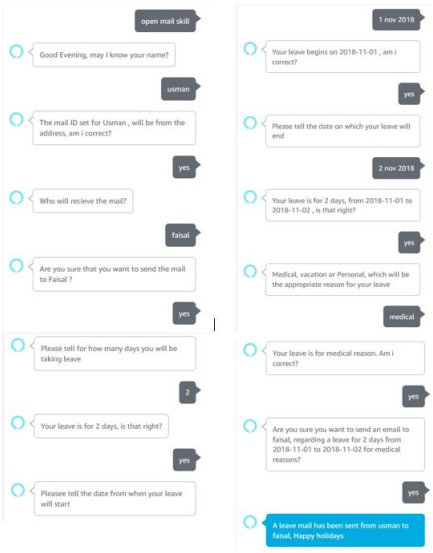
(b) Event generated in AWS CloudWatch for *IT Maintenance Skill*

(c) Slack output of *IT Maintenance Skill*

Figure 16: Testing of *IT Maintenance Skill*

6.2 Testing of IT Maintenance Skill

This skill has gone through the same testing phase as by *Room Lamp Skill*, however, its output can be verified from the message received in the slack channel. The testing phase of *IT Maintenance Skill* is shown in Figure 16. As the dialog model is included for IT Maintenace skill, therefore Alexa has to continue conversation with user until all intents are acquired. Everytime Alexa asks for user input, a response is



(a) Input given to the *Mail Skill* using Alexa developer test simulator

(b) Event generated in AWS CloudWatch for *Mail Skill*

(c) Mail output of *Mail Skill*

Figure 17: Testing of *Mail Skill*

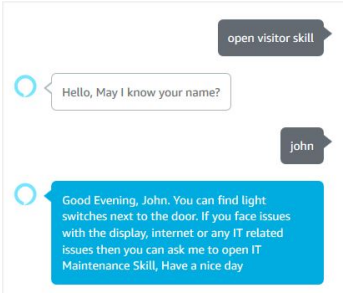
generated which can be seen in Figure 16(b). After completion of the dialog model is confirmed, Final response is generated which can be seen in Figure 16(b) and output message is received in the slack channel as shown in Figure 16(c).

6.3 Testing of Mail Skill

The complete testing phase of *Mail Skill* is shown in Figure 17. The dialog model is a part of task execution therefore series of responses triggered to obtain all the details from the user that are required to send an email, which can be seen in Figure 17(b). The output of this skill is the leave mail, which is to be received by the receiver mentioned in the Figure 17(a) and received mail is shown in Figure 17(c).

6.4 Testing of Visitor Skill

The testing phase of *Visitor skill* is presented in Figure 18.The final output of this skill is only a prompt message from Alexa therefore it only requires Alexa developer test simulator and AWS CloudWatch for testing.



(a) Input given to the *Visitor Skill* using Alexa developer test simulator

(b) Event generated in AWS CloudWatch for *Visitor Skill*

Figure 18: Testing of *Visitor Skill*

6.5 Discussion

The functionality of 'Alexa Office' is mainly dependent upon the working of each skill. As discussed in Section 4.2, each skill is based on three sections; Interaction Model, Backend Services and Thrid party app service. Each section is individually designed and then integrated with each other. This makes the whole system scalable and flexible. Any part of skill can be modified or updated any time without making changes in system architecture. Each skill has tested in terms of functionality, correct behavior and the dialog model.

As mentioned in Section 5.2, the purpose of room lamp skill was to establish the proof of concept of an integration between Alexa and any physical device. The main

challenge in the skill was to enable communication between device and Alexa cloud and other challenges included task execution capability and dialog model. The testing results presented in Section 6.1 shows that Room skill is completely functional and exhibit correct behavior towards user input. However, system operation is quite simple as it just needs to change the power status of physical device and it does not require any additional information except desired power status of device, the dialog model has kept quite simple with single intent.

Other skills were developed to establish the proof of concept that UI being part of the building system can execute non productive tasks while user can do other productive tasks at the same time. Three skill were suggested in Section 4.1, these skills were developed in a same way as room lamp skill as each layer is individually developed and then integrating to obtain correct behavior of skill. These skills were tested and results were presented in Sections 6.2,6.3 and 6.4. The purpose of IT Maintenance Skill was to send IT related complaints to IT department on behalf of user. The results shown in Section 6.2, this skill is functionally capable of sending complaints to IT department through the slack channel. Moreover, the dialog model is included to improve the dialog model of this skill during the development phase in order to collect information from the user. The improved dialog model makes it more usable and enhances ease of use. However, the dialog model can be further improved to increase usability without changing in skill architecture. From the successful testing of IT Maintenance skill, it is evident that this UI is capable of executing user's message to different stake holders by using any messaging tool such as slack, which is used in this project.

Mail skill is developed to establish the proof of concept that UI is capable of sending emails on user's behalf. The idea of Mail Skill, which was discussed in Section 5.4 is developed and tested. The results presented in Section 6.3 prove the capability of UI to send emails. Additionally, a simple skill named "Visitor skill" is developed and tested as shown in Section 6.4. This skill is only based on dialog model, therefore there is no third-party API is used with the skill. The purpose of visitor skill as discussed in 5.5 was to give general instructions to the visitor who visits the room so that visitor can be aware of room's system functions.

Considering the functionality and purpose of each skill, improvements could be easily made to the interaction model, this however is a minor retouch and could be improved to enhance usability and ease of use of the user. These skills are deployed in Alexa Office and are available to its user as shown in Figure 14. Therefore, in evidence of Chapter 5 and 6, the functional requirement of 'Alexa Office' mentioned in Section 4.1 is successfully achieved.

7 Conclusion

The main outcomes of this thesis were the selection of the best possible user centered based UI type based on comparative analysis and development of a UI prototype that can determine its feasibility of integration with existing BAS of the office building. The literature review discussed the work related to the evolution of UI designs with BAS, evolution parameters that can be used to make comparison between existing systems and then selected the best possible user building interaction mode from this comparative analysis. The Amazon Alexa platform is used for the development of UI prototype for the office building. The developed solution then tested and verified using simulators provided by AWS and Amazon Alexa.

In the first part of the thesis, detail study was conducted to analyze different modes of user building interaction and how UI designs had been evolved for BAS. On the basis of background work and comparative analysis of existing systems, it was deduced that all developed UI design types can offer flexibility, portability, remote accessibility, usability but voice recognition is the only feature that offers more system acceptability to all age group that most of the existing systems lacked. The Amazon Alexa, a virtual assistant is selected as the development platform that is more system adaptable in comparison to other virtual assistants.

In second part of the thesis, a proof of concept is developed by using Alexa device, Alexa skills, Alexa for Business, Amazon Web services and third party APIs and thereby, four skills were implemented. All skills developed based on custom skills and AWS Lambda as endpoint hosting. In addition, device cloud is developed by using AWS IOT that connects with Raspberry Pi3 that worked as device controller for room lamp skill. Amazon SES is used for mail services whereas incoming webhook is used to post messages on the slack. Interaction model is developed using the Alexa console and it is further connected to endpoints provided by AWS Lambda. AWS Lambda then calls task execution functions in respective skill operation after acquiring all required intent slot values.

In room lamp skill, executable function sends device update status to AWS IOT, which onward sends command to Raspberry Pi3 for action. In IT Maintenance skill, the purpose of executable function to post a message to the slack channel via HTTP

provided by incoming webhook. Whereas in mail skill, the purpose of executable function is to send the details of email to the Amazon SES which onward sends email to the recipient. Visitor skill does not require any executable action. On completion of the procedure, which can be either successful or unsuccessful, AWS Lambda generates a JSON response and sends it to the Alexa device. Alexa device then prompts the corresponding message to the user. The developed interaction model also uses the dialog model specifically for IT maintenance skill and Mail skill to improve the correctness of task execution.

Although developed solution is technically feasible to integrate with existing BAS however, the selected platform possesses certain regional limitations. A4B is only available in the US-east region as well as AWS is also not available all over Europe. Therefore, all skills have used US-east region data center for data storage. This presents two major drawbacks; one is latency and other one is data governance. As all data is stored in the US east region and user is in Europe, it causes significant delay in sending and receiving data. Other issue is the data management, which requires legal permission to allow data to be stored in the data center outside of Europe. In addition to this, A4B is not free tier. It charges $3 per month for each enrolled user and $7 for each shared device.

The proposed solution is developed for the office building that exhibits multiple users and multiple rooms inside the building. This suggests that it can be replicated for hospital buildings or school buildings that may contribute in improving healthcare facilities and education standard respectively. The three layer approach has given scalability to the system therefore, interaction model can be made more usable and useful by conducting different empirical tests. More skills can be developed for multiple electrical appliances using the same approach as used in room lamp skill. Other than skills related to BAS, more skills can be developed using the same platform that can help in user productivity at work and enhancing system's performance.

References

[1] W. F. Preiser and U. Schramm, "Intelligent office building performance evaluation," *Facilities*, vol. 20, no. 7/8, pp. 279–287, 2002.

[2] M. Wigginton and J. Harris, *Intelligent Skins*. Routledge, 2002.

[3] J.K.W.Wong, H.Li, and S.W.Wang, "Intelligent building research: a review," *Automation in Construction*, vol. 14, no. 1, pp. 143–159, 2005.

[4] C. C. Federspiel and L. Villafana, "Design of an energy and maintenance system user interface for building occupants," *ASHRAE*, vol. 109, pp. 656–676, 2003.

[5] A. Feige and H. Wallbaum, "Impact of sustainable office buildings on occupant's comfort and productivity," *Journal of Corporate Real Estate*, vol. 15, pp. 7–34, 2013.

[6] MonikaFrontczak and PawelWargocki, "Literature survey on how different factors influence human comfort in indoor environments," *Building and Enviornment*, vol. 46, no. 4, pp. 922–937, 2011.

[7] D. Leonard-Barton and D. K. Sinha, "Developer-user interaction and user satisfaction in internal technology transfer," *The Academy of Management Journal*, vol. 36, no. 5, pp. 1125–1139, 1993.

[8] C. Z. Yue and S. Ping, "Voice activated smart home design and implementation," in *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, April 2017, pp. 489–492.

[9] J. Ketomäki and H. Ihasalo. Pilot project for smart building services technology investigates the functionality of building services technology in the future in an everyday environment. [Online]. Available: https://www.aalto.fi/news/pilot-project-for-smart-building-services-technology-investigates-the-functionality-of

[10] W. V. Hans-Joachim Platte Gunter Oberjatzas, "Remote control device for controlling various functions of one or more appliances," Google Patents, 1988, uS Patent 4,728,949.

[11] Y. Fujita and S. P. Lam, "Distribution system and method for menu-driven user interface," Google Patents, 1996, uS Patent 5,500,794.

[12] M. Stein, T. R. Kaufman, Y. A. Richarz, K. A. Tarlow, and B. C. Nesbitt, "User interface for home automation system," Google Patents, 2000, uS Patent 6,140,987.

[13] F. Papai, A. Zoldi, and P. M. Corcoran, "User interface technologies for home appliances and networks," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 679–685, 1998.

[14] A.-R. Al-Ali and M. Al-Rousan, "Java-based home automation system," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 498–504, 2004.

[15] M. V. D. Werff, X. Gui, and W. Xu, "A mobile-based home automation system," in *2005 2nd Asia Pacific Conference on Mobile Technology, Applications and Systems*, 2005.

[16] R. Shahriyar, E. Hoque, S. Sohan, I. Naim, M. M. Akbar, and M. K. Khan, "Remote controlling of home appliances using mobile telephony," *International Journal of Smart Home*, vol. 2, no. 3, pp. 37–54, July 2008.

[17] R. Piyare, "Internet of things: ubiquitous home control and monitoring system using android based smart phone," *International Journal of Internet of Things*, vol. 2, no. 1, pp. 5–11, 2013.

[18] J. Nielsen and S. Gilutz, *Usability return on investment.* Nielson Norman Group, 2003.

[19] X. Zeng, A. Fapojuwo, and R. Davies, "Design and performance evaluation of voice activated wireless home devices," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 3, pp. 983–989, 2006.

[20] H. AlShu'eili, G. S. Gupta, and S. Mukhopadhyay, "Voice recognition based wireless home automation system," in *Mechatronics (ICOM), 2011 4th International Conference On.* IEEE, 2011, pp. 1–6.

[21] G. Muthuselvi and S. B., "Real time speech recognition based building automation system," *ARPN Journal of Engineering and Applied Sciences*, vol. 9, no. 18, pp. 5015–5028, 2014.

[22] A. Pandey, U. Mishra, and A. Chaubey, "Voice controlled home automation," *International Journal of Research In Science & Engineering*, pp. 89–91, March 2017.

[23] S. H. Baria and C. Bhatt, "Personal and intelligent home assistant to control devices using raspberry pi," *International Journal of Computing and Digital Systems*, vol. 6, no. 4, pp. 213–220, July 2017.

[24] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface: Talking to Smart Devices*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[25] E. Tatjana, I. Sandra, M. Sunčica, M. Milica, and S. Nikola, "Voice control for smart home automation: Evaluation of approaches and possible architectures," in *Consumer Electronics-Berlin (ICCE-Berlin), 2017 IEEE 7th International Conference on*, 2017, pp. 140–142.

[26] M. B. Hoy, "Alexa, siri, cortana, and more: An introduction to voice assistants," *Medical reference services quarterly*, vol. 37, no. 1, pp. 81–88, 2018.

[27] Amazon. Amazon alexa. [Online]. Available: https://developer.amazon.com/alexa.

[28] Interaction model. [Online]. Available: https://developer.amazon.com/docs/alexa-voice-service/interaction-model.html

[29] Dialog model. [Online]. Available: https://developer.amazon.com/docs/custom-skills/dialog-interface-reference.html

[30] Understand custom skills. [Online]. Available: https://developer.amazon.com/docs/custom-skills/understanding-custom-skills.html#host-the-cloud-based-service-for-your-skill

[31] Host a custom skill as an aws lambda function. [Online]. Available: https://developer.amazon.com/docs/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html

[32] Aws cloud watch. [Online]. Available: https://aws.amazon.com/cloudwatch/

[33] S. API. Incoming webhooks. [Online]. Available: https://api.slack.com/incoming-webhooks

[34] Amazon ses. [Online]. Available: https://docs.aws.amazon.com/ses/latest/DeveloperGuide/Welcome.html

Appendix A:
Complete Summary of existing systems.

Loxone	
System Features	
HVAC	✓
Lighting	✓
Security	✓
Music	✓
Window Shading	✓
User Adaptability	
Flexibility	Task Scheduling, Custom Scenes, Pre-Set Scenarios
Privacy	IP Connectivity, Require Username and Password
UI Customization	✓
Functionality	
Remote Access	✓
Adaptability	✓
Portability	✓
Accessibility	Only visual modality
Control4	
System Features	
HVAC	✓
Lighting	✓
Security	✓
Music	✓
Window Shading	✓
User Adaptability	
Flexibility	Task Scheduling, Custom Scenes, Pre-Set Scenarios
Privacy	IP Connectivity, Require Username and Password for first time only
UI Customization	✗
Functionality	
Remote Access	✓
Adaptability	✗
Portability	Only works on iOS
Accessibility	Only visual modality

Zipato	
System Features	
HVAC	✓
Lighting	✓
Security	✓
Music	✗
Window Shading	✗
User Adaptability	
Flexibility	Task Scheduling, Custom Scenes
Privacy	IP Connectivity, Session based login
UI Customization	✗
Functionality	
Remote Access	✓
Adaptability	✗
Portability	Only works on iOS and Web
Accessibility	Only visual modality
Crestron	
System Features	
HVAC	✓
Lighting	✓
Security	✓
Music	✗
Window Shading	✓
Functionality	
Flexibility	Custom Scenes
Privacy	IP Connectivity
UI Customization	✗
Functionality	
Remote Access	✓
Adaptability	✗
Portability	Only works on iOS
Accessibility	✓
Vivint	
System Features	
HVAC	✓
Lighting	✓

Security	✓
Music	✗
Window Shading	✓
Functionality	
Flexibility	Custom Scenes, Pre-Set Scenarios
Privacy	Session based login
UI Customization	✗
Functionality	
Remote Access	✓
Adaptability	✗
Portability	✓
Accessibility	Only visual modality